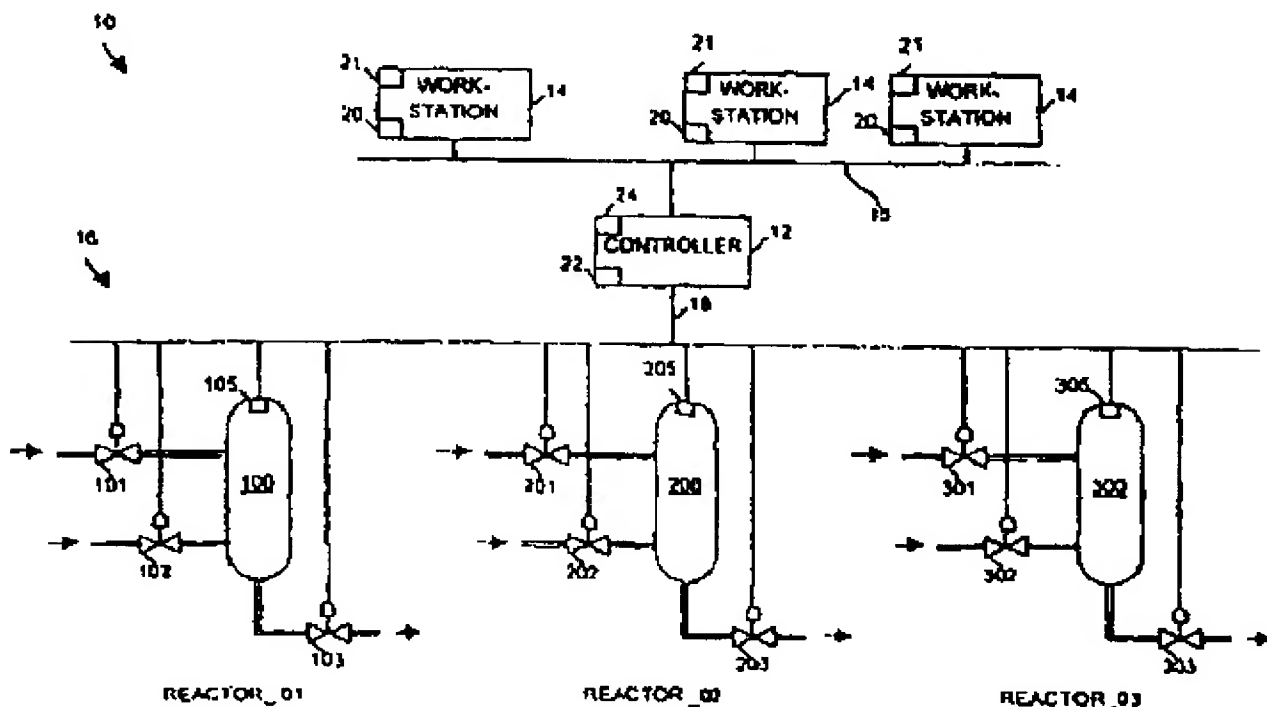


AN: PAT 2000-567207
 TI: Process control system with generic control routine which creates instantiated control programs
 PN: GB2348020-A
 PD: 20.09.2000
 AB: NOVELTY - The process control system (10) includes a controller (12) coupled to numerous workstations (14) via an Ethernet connection (15). The controller is capable of communicating with control elements, such as field devices and function blocks, within field devices distributed throughout the process (16) to perform one or more process control routines to implement desired control of the process. The control system implements batch processes that operate reactor units which create a product e.g. food, drugs. DETAILED DESCRIPTION - An independent claim is included for a software control component for use in a process control system; USE - Uses indirect referencing in process control routines to enable advanced process control ADVANTAGE - Enables indirect referencing using alias names and dynamic reference parameters DESCRIPTION OF DRAWING(S) - Partial block diagram of process control network Control system 10 Controller 12 Workstations 14 Ethernet 15 Process 16
 PA: (ROEC) FISHER-ROSEMOUNT SYSTEMS INC;
 IN: DEITZ D L; HAVEKOST R B; IRWIN W G; STEVENSON D L;
 FA: GB2348020-A 20.09.2000; GB2348020-B 24.12.2003;
 DE10011661-A1 14.09.2000; JP2000311004-A 07.11.2000;
 US6385496-B1 07.05.2002;
 CO: DE; GB; JP; US;
 IC: G05B-019/02; G05B-019/042; G05B-019/418; G05B-019/42;
 G06F-019/00; H04L-012/40;
 MC: T01-J; T06-A04A2A; T06-A04B7;
 DC: T01; T06;
 FN: 2000567207.gif
 PR: US0267431 12.03.1999;
 FP: 14.09.2000
 UP: 15.01.2004





18 BUNDESREPUBLIK
DEUTSCHLAND



DEUTSCHES
PATENT- UND
MARKENAMT

12 Offenlegungsschrift
10 DE 100 11 661 A 1

51 Int. Cl. 7:
G 05 B 19/042

21 Aktenzeichen: 100 11 661.2
22 Anmeldetag: 10. 3. 2000
43 Offenlegungstag: 14. 9. 2000

1 US 6,385,496 B1

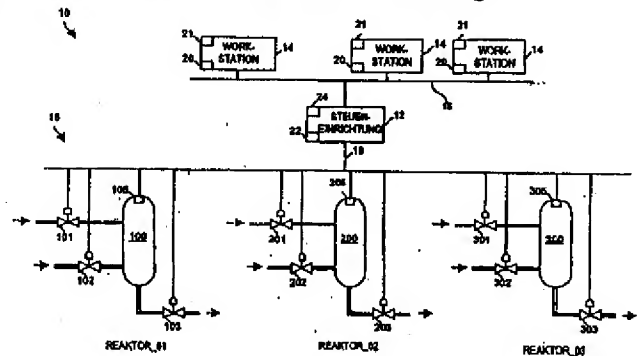
DE 100 11 661 A 1

30 Unionspriorität:
09/267,431 12. 03. 1999 US
71 Anmelder:
Fisher-Rosemount Systems., Inc.
(n.d.Ges.d.Staates Delaware), Austin, Tex., US
74 Vertreter:
Meissner, Bolte & Partner, 80538 München

72 Erfinder:
Irwin, William G., Austin, Tex., US; Havekost,
Robert B., Austin, Tex., US; Stevenson, Dennis L.,
Round Rock, Tex., US; Deitz, David L., Austin, Tex.,
US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

- 54 Prozeßsteuersystem mit Prozeßsteuerroutinen unter Verwendung von indirekter Referenzierung
57 Prozeßsteuersystem (10), in dem eine generische Prozeßsteuerroutine geschrieben wird, die Aliasnamen und dynamische Referenzparameter enthält. Vor der Ausführung einer Prozeßsteuerfunktion an einer bestimmten Einheit des Prozeßsteuersystems wird eine Instanz der generischen Routine erstellt, in der die Aliasnamen durch Parameter ersetzt werden, die in einer Alias-Auflösungstabelle für die bestimmte Einheit definiert sind. Die Steuereinrichtung (12) führt anschließend die diesbezüglich konkretisierte Version der generischen Routine aus, um den Betrieb der Einheit zu steuern. Die generische Routine kann mehrere Algorithmen haben, die zu dieser gehören, wobei jeder Algorithmus so gestaltet ist, daß er unterschiedliche Einheiten steuert, die geringfügig verschiedene Hardware haben, wenngleich diese verschiedenen Einheiten im wesentlichen dieselbe Funktion innerhalb des Prozeßsteuersystems (10) ausführen. Die generische Routine kann ferner mit mehreren Klassen von Hardware verwendet werden, die unterschiedliche Funktionen innerhalb des Prozeßsteuersystems ausführen. Die dynamischen Referenzparameter der generischen Prozeßsteuerroutine ermöglichen es, daß ein Feld bei oder während der Laufzeit spezifiziert wird.



DE 100 11 661 A 1

Die vorliegende Erfindung betrifft allgemein Prozeßsteuernetze und insbesondere die Verwendung von indirekter Referenzierung in Prozeßsteuer Routinen, um eine fortschrittliche Prozeßsteuerung zu ermöglichen.

Prozeßsteuernetze, wie beispielsweise die in chemischen Prozessen, in Erdölverarbeitungsprozessen oder anderen Prozessen verwendeten, enthalten allgemein eine zentrale Prozeßsteuereinrichtung, die mit einer oder mehreren Anlageneinrichtungen in Kommunikationsverbindung steht, bei denen es sich beispielsweise um Ventilpositioniereinrichtungen, Schalter, Sensoren (wie z. B. Temperatur-, Druck- und Durchflußmengensensoren) etc. handeln kann. Diese Anlageneinrichtungen können physische Steuerfunktionen innerhalb des Prozesses (wie z. B. das Öffnen und das Schließen eines Ventils) durchführen, können Messungen innerhalb des Prozesses zur Verwendung bei der Regelung des Betriebsablaufes des Prozesses vornehmen oder können jede andere gewünschte Funktion innerhalb des Prozesses ausführen.

Prozeßsteuereinrichtungen waren bisher über eine oder mehrere analoge Signalleitungen oder Busleitungen mit Anlageneinrichtungen verbunden, welche beispielsweise 4-20 mA-Signale zu und von den Anlageneinrichtungen leiten können. In jüngerer Zeit hat die Prozeßsteuerindustrie jedoch eine Anzahl von standardisierten, offenen digitalen oder kombiniert digitalen und analogen Kommunikationsprotokollen entwickelt, wie z. B. das FOUNDATION™-FIELD BUS (nachfolgend als "Fieldbus" bezeichnet), HART®, PROFIBUS®, WORLD FIP®, Device-Net® und CAN-Protokoll, welche verwendet werden können, um die Kommunikation zwischen einer Steuereinrichtung und Anlageneinrichtungen umzusetzen. Allgemein ausgedrückt empfängt die Prozeßsteuereinrichtung Signale, welche Messungen, die von einer oder mehreren Anlageneinrichtungen durchgeführt wurden, und/oder andere Informationen darstellen, die zu den Anlageneinrichtungen gehören, verwendet diese Informationen, um eine typischerweise komplexe Steueroutine umzusetzen und erzeugt Steuersignale, die über die Signalleitungen oder Busleitungen zu den Anlageneinrichtungen gesendet werden, um dadurch den Betriebsablauf des Prozesses zu steuern.

Bestimmte Arten von Prozeßsteuernetzen, wie beispielsweise diejenigen, die in Stapelprozessen verwendet werden, enthalten typischerweise mehrere Garnituren von parallel angeordneten Geräten, die so gestaltet sind, daß sie dieselbe oder eine ähnliche Ausrüstung haben, welche im wesentlichen dieselbe Funktion innerhalb der Prozesse ausführt. Somit kann beispielsweise eine Fabrik zur Herstellung von Keksen mehrere Garnituren von Mischgeräten, mehrere Garnituren von Backgeräten und mehrere Garnituren von Verpackungsgeräten haben, wobei alle einzelnen Mischgeräte in der Lage sind, parallel zu arbeiten und so miteinander verbunden werden können, daß sie in Reihe mit jedem der Backgeräte und jedem der Verpackungsgeräte arbeiten. In einem derartigen System ist es anstrengenswert, in der Lage zu sein, den selben Steueralgorithmus oder dieselbe Steueroutine zu verwenden, um den Betrieb jeder bestimmten Garnitur von parallel vorhandenen Geräten zu steuern, um dadurch die Anzahl der Steuer Routinen zu reduzieren, die erstellt und innerhalb der Steuereinrichtung gespeichert werden müssen. Diese Steuer Algorithmen müssen jedoch so geschrieben werden, daß sie bei der Ausführung die Ausrüstung einer bestimmten Einheit, die zu dieser Zeit verwendet wird, spezifizieren.

Bei einigen Systemen nach dem Stand der Technik wurde eine generalisierte Steueroutine, welche Aliasnamen (das heißt nicht spezifizierte Variable) verwendete, um das spezifische Gerät zu bezeichnen, das von Paralleleinheit zu Paralleleinheit unterschiedlich war, in einer Workstation geschaffen. Um eine Systemsteuereinrichtung in die Lage zu versetzen, die generalisierte Steueroutine auf einer bestimmten Einheit auszuführen, wurde das generalisierte Programm unter Verwendung einer Alias-Auflösungstabelle, die für eine bestimmte Einheit geschaffen wurde, instantiiert. Eine derartige Alias-Auflösungstabelle enthielt eine Definition für jeden Aliasnamen, der in der generalisierten Steueroutine verwendet wird, und wurde verwendet, um eine ausführbare Instanz der generalisierten Steueroutine durch Substituieren der Werte in der Alias-Auflösungstabelle der bestimmten Einheit durch die Aliasnamen in der Steueroutine zu schaffen. Diese diesbezüglich konkretisierte Steueroutine wurde anschließend in die Steuereinrichtung heruntergeladen und in dieser gespeichert und daraufhin während der Laufzeit verwendet, um einen Steuervorgang (oder eine Steuerphase) in der bestimmten Einheit durchzuführen. Unter Verwendung dieses System mußte jedoch die Steuereinrichtung eine separate diesbezüglich konkretisierte (aufgelöste) Steueroutine für jede der unterschiedlichen Paralleleinheiten speichern, was eine große Menge von Speicherplatz in der Steuereinrichtung erforderlich machte, insbesondere dann, wenn die Steuereinrichtung dafür verwendet wurde, eine große Anzahl von ähnlichen Einheiten zu steuern, und verwendet wurde, eine große Anzahl von unterschiedlichen Operationen oder Phasen in jeder Einheit auszuführen (da separate diesbezüglich konkretisierte Steuer Routinen für jede Phase für jede Einheit erforderlich waren).

Bei einem anderen System nach dem Stand der Technik wurde die generalisierte Steueroutine in der Steuereinrichtung gespeichert und während der Laufzeit verwendet, um die programmierte Operation oder Phase in einer der Paralleleinheiten, auf die sie angewandt wurde, durchzuführen. In diesem Fall wurden die Aliasnamen während der Laufzeit unter Verwendung der Alias-Auflösungstabelle für die bestimmte Einheit, die gesteuert wurde, fliegend aufgelöst. Bei dieser Konfiguration mußte jedoch dann, wenn eine Veränderung an der generalisierten Steueroutine, welche gegenwärtig durch die Steuereinrichtung gefahren wurde, durchzuführen war, der Lauf dieser Routine unterbrochen werden, um es zu ermöglichen, daß eine neue generalisierte Steueroutine in die Steuereinrichtung heruntergeladen wird. Dies führte zu Verlusten an Zeit, Material etc. im Zusammenhang mit dem abgebrochenen Lauf des Prozesses.

Ferner ermöglichte es keines dieser bekannten Systeme, eine einzelne generische Prozeßsteueroutine mit Aliasnamen über mehrere oder unterschiedliche Klassen von Einheiten oder Geräten anzuwenden. Tatsächlich war bei diesen Systemen nach dem Stand der Technik eine Steueroutine für eine Phase auf die Verwendung mit einer Einheitenklasse, das heißt einer bestimmten Art von Hardwareeinheit, wie z. B. Reaktoren oder Mischer etc., beschränkt. Als Resultat mußte zunächst eine generische Prozeßsteueroutine geschaffen und gespeichert werden, um beispielsweise ein Reaktorgefäß zu füllen, während eine zweite generische Prozeßsteueroutine geschaffen und gespeichert werden mußte, um Misch-tanks zu füllen, und eine dritte geschaffen und gespeichert werden mußte, um Zuliefertanks zu füllen, was zu der Schaffung von vielen unterschiedlichen generischen Steuer Routinen zum Ausführen von im wesentlichen derselben Funktion an unterschiedlichen Arten von Hardware führte.

Entsprechend ermöglichte es keines dieser Systeme nach dem Stand der Technik der generischen Steueroutine, Unterschiede zwischen den Geräten zu berücksichtigen, die zu den unterschiedlichen Modulen einer bestimmten Art einer

Hardwareeinheit gehören. Wenn beispielsweise eine erste Reaktor-Einheit ein elektrisches Heizelement hatte und eine zweite Reaktor-Einheit ein Dampfheizelement, mußte eine unterschiedliche generische Steueroutine zum Aufheizen jeder dieser Reaktor-Einheiten entwickelt werden, um die Unterschiede bei der Steuerung der elektrischen Heizung und der Dampfheizung zu berücksichtigen, obgleich der Prozeß nur erforderte, daß der Aufheizvorgang ausgeführt wurde, wobei die Art der Heizung irrelevant war. Dieses Problem tritt allgemein auf, wenn zusätzliche Einheiten oder Module (wie z. B. Reaktormodule) zu verschiedenen Zeiten zu einem Prozeßsteuersystem zugeführt werden und aus Gründen von Kosten, kürzlichen Fortschritten hinsichtlich der Hardware etc., die neu hinzugefügten Module, während sie so konstruiert sind, daß sie im wesentlichen dieselbe Funktion wie die vorhandenen Module ausführen, mit geringfügig anderen Geräten ausgerüstet sind.

Ferner hatten diese Systeme nach dem Stand der Technik kein einfaches Verfahren zum Spezifizieren eines Parameters, der während der Laufzeit der Prozeßsteueroutine für eine Phase eines Prozesses identifiziert werden sollte. Tatsächlich wurde bei den meisten Systemen nach dem Stand der Technik, die indirekte Referenzierung aufwiesen, die Alias-Auflösungstabelle verwendet, um Aliasnamen aufzulösen, wenn die Prozeßsteueroutine konfiguriert wurde und in maschinenlesbaren Code umgesetzt wurde, was vor der Laufzeit erfolgte. Um es zu ermöglichen, daß eine Variable während der Laufzeit geändert oder spezifiziert wurde, war bei einem System nach dem Stand der Technik ein Adressierschema, wie z. B. ein Adressenarray, vorgesehen, in welchem Referenzen oder Zeiger während der Laufzeit platziert werden konnten, so daß dann, wenn das Steuerprogramm eine Instruktion erreichte, die auf eine der Adressen in dem Array verwies, das Programm zu der Einrichtung oder Position gehen würde, auf die durch den Inhalt der angegebenen Adresse verwiesen wurde. Es gab jedoch keine Möglichkeit, anzugeben, ob der Inhalt der Verweisadresse auf eine gültige Einrichtung oder eine ordnungsgemäße Position innerhalb der Steueroutine während der Laufzeit verwies. Wenn der Zeiger ungültig war, wird das Programm unterbrochen und nicht fortgeführt, was zu einem Produktionsstopp führt. Ferner war dieses Adressierschema komplex und schwierig in ordnungsgemäßer Weise zu verwenden, da es ein detailliertes Wissen über das Adressenarray, welches den Zeiger enthielt, und Kenntnisse, welche Adresse des Array von der Steueroutine zu welcher Zeit verwendet wurde, erforderte. Somit war es sehr arbeitsaufwendig seitens der Programmierer und der Benutzer, sicherzustellen, daß ein korrekter Zeiger an der korrekten Adresse zur korrekten Zeit gespeichert war, um die Unterbrechung der Steueroutine während der Laufzeit zu verhindern.

Es ist Aufgabe der Erfindung, ein Prozeßsteuersystem zu schaffen, bei dem die nach dem Stand der Technik auftretenden Probleme vermieden werden.

Die Lösung der Aufgabe ergibt sich aus den Patentansprüchen. Unteransprüche beziehen sich auf bevorzugte Ausführungsformen der Erfindung, wobei auch andere Kombinationen von Merkmalen als in den Ansprüchen beansprucht möglich sind.

Ein Prozeßsteuersystem enthält eine oder mehrere Prozeßsteueroutinen, die eine indirekte Referenzierung unter Verwendung von Aliasnamen und/oder dynamischen Referenzparametern ermöglichen. Eine generische Prozeßsteueroutine wird so geschrieben, daß sie Aliasnamen enthält, und diese generische Prozeßsteueroutine wird in einer Steuereinrichtung gespeichert, die einen Prozeß steuert, der beispielsweise parallel vorhandene Ausrüstungen (parallel vorhandene Einheiten) aufweist. Vor der Ausführung einer Prozeßsteuerfunktion an einer bestimmten Einheit wird eine Instanz der generischen Routine, die diese Funktion steuert, geschaffen, in welcher die Aliasnamen in der generischen Steueroutine durch Parameter ersetzt werden, die in einer Alias-Auflösungstabelle für die bestimmte Einheit definiert sind. Die Steuereinrichtung führt dann die diesbezüglich konkretisierte Version der generischen Routine aus, um den Betriebsablauf der Einheit zu steuern. Dies reduziert die Speicheranfordernisse der Steuereinrichtung, da es der Steuereinrichtung erlaubt, nur die generische Routine und die instantiierten Versionen dieser Routine zu speichern, die gegenwärtig laufen, anstatt daß eine diesbezüglich konkretisierte Version der generischen Routine für alle Einheiten einzeln zu allen Zeiten gespeichert wird. Ferner ermöglicht dies, daß die generische Steueroutine verändert wird, während eine Instanz derselben ausgeführt wird, ohne daß die in Ausführung befindliche Routine abgebrochen werden muß.

Falls erwünscht, können mit dem generischen Programm mehrere Algorithmen verbunden sein, wobei jeder der unterschiedlichen Algorithmen so gestaltet ist, daß er verschiedene Einheiten steuert, die geringfügig unterschiedliche Hardware haben, auch wenn diese unterschiedlichen Einheiten im wesentlichen dieselbe Funktion innerhalb des Prozeßsteuersystems ausführen. Wenn eine diesbezüglich konkretisierte Version des generischen Programms geschaffen wird, bestimmt die Steuereinrichtung, welcher der mehrfachen Algorithmen der generischen Routine zu verwenden ist, und zwar auf der Basis einer gespeicherten Angabe, welche die Hardwarekonfiguration der bestimmten Einheit identifiziert, für welche die diesbezüglich konkretisierte Steueroutine geschaffen wird. Das System ermöglicht es ferner, eine Steueroutine für mehrere Einheitenklassen oder unterschiedliche Arten von Hardware zu schaffen und an diesen anzuwenden, welche verwendet werden, um unterschiedliche Funktionen innerhalb des Prozeßsteuersystems auszuführen. In diesem Fall kann eine Konfigurationsroutine sicherstellen, daß eine Alias-Definition für jeden Aliasnamen in jeder Alias-Auflösungstabelle existiert, die zu den unterschiedlichen Klassen von Einheiten gehört, an welchen die generische Prozeßsteueroutine angewendet werden soll.

Dies ermöglicht es, daß weniger generische Steueroutinen geschrieben und in der Steuereinrichtung gespeichert werden, da eine einzelne generische Steueroutine geschrieben und verwendet werden kann, um eine bestimmte Funktion an verschiedenen Arten von Geräten auszuführen, die für unterschiedliche Zwecke innerhalb des Prozeßsteuersystems verwendet werden.

Darüber hinaus kann die Prozeßsteueroutine einen oder mehrere dynamische Referenzparameter verwenden, um es zu ermöglichen, daß ein Feld, eine Einrichtung, ein Parameter, etc. spezifiziert werden, nachdem das diesbezüglich konkretisierte ausführbare Programm erstellt wurde, das heißt, es zu ermöglichen, daß dynamisch ein Feld referenziert wird, und zwar zu der Zeit oder während der Zeit des Laufes. Der dynamische Referenzparameter hat mehrere Attribute, darunter beispielsweise ein Referenzattribut, das einen Zeiger, einen Pfad oder ein Identifizierungskennzeichen für die Einrichtung, das Feld bzw. den Parameter etc., die bzw. das referenziert wird, speichert, und ein Verbindungsattribut, das kennzeichnet, ob eine tatsächliche Verbindung zu dem Feld, das von dem Referenzattribut angegeben wurde, hergestellt werden kann, beispielsweise ob das Referenzattribut ein gültiges Feld innerhalb der Prozeßsteuersystemkonfiguration

definiert. Der dynamische Referenzparameter kann ferner Attribute enthalten, die das Lesen aus und/oder Schreiben in das durch das Referenzattribut spezifizierten Feld als ein String oder als ein numerischer Wert erlauben. Ferner kann der dynamische Referenzparameter ein oder mehrere Attribute einschließen, die das Lesen eines oder mehrerer Statuswerte erlauben, die zu dem durch das Referenzattribut spezifizierten Feld gehören, darunter beispielsweise der Status dieses Feldes und der Status des letzten Schreibvorganges in dieses Feld.

5 Nachfolgend wird eine Ausführungsform der Erfindung unter Bezug auf die Zeichnung näher erläutert.

Fig. 1 zeigt eine teilweise als Blockdiagramm und teilweise als schematisches Diagramm ausgeführte Darstellung eines Steuernetzes, das eine oder mehrere Steuerroutinen verwendet, die Aliasnamen und/oder dynamische Referenzparameter zur Ausführung der Steuerung von Prozeßgeräten haben;

10 Fig. 2 ist ein Blockdiagramm einer Objektstruktur, die eine konzeptuelle Konfiguration des Prozeßsteuernetzes von Fig. 1 zeigt; und

Fig. 3 ist ein erweitertes Blockdiagramm eines Abschnitts der Objektstruktur von Fig. 2.

Wie Fig. 1 zeigt, enthält ein Prozeßsteuernetz 10 eine Prozeßsteuereinrichtung 12, die mit zahlreichen Workstations 14 über eine Ethernet-Verbindung 15 verbunden ist. Die Steuereinrichtung 12 ist ferner mit Einrichtungen oder Geräten innerhalb eines Prozesses (allgemein mit Bezugszeichen 16 bezeichnet) über eine Gruppe von Kommunikationsleitungen oder einen Bus 18 verbunden. Die Steuereinrichtung 12, die beispielsweise eine DeltaVTM-Steuereinrichtung sein kann, die von Fisher-Rosemont-Systems, Inc. vertrieben wird, ist in der Lage, mit den Steuerelementen, wie z. B. Anlageneinrichtungen und Funktionsblöcken in Anlageneinrichtungen, zu kommunizieren, die über den Prozeß 16 verteilt sind, um eine oder mehrere Prozeßsteuerroutinen durchzuführen und dadurch die gewünschte Steuerung des Prozesses 16 umzusetzen. Die Workstations 14 (welche beispielsweise Personalcomputer sein können) können von einem oder mehreren Technikern oder Benutzern verwendet werden, um Prozeßsteuerroutinen zu gestalten, die von der Steuereinrichtung 12 auszuführen sind, um mit der Steuereinrichtung 12 zu kommunizieren, um derartige Prozeßsteuerroutinen herunterzuladen, und um Informationen zu empfangen und darzustellen, die zu dem Prozeß 16 während des Betriebsablaufs des Prozesses 16 gehören. Jede der Workstations 14 enthält einen Speicher 20 zum Speichern von Anwendungsprogrammen, wie z. B. Konfigurationsgestaltungsanwendungen, und zum Speichern von Daten, wie z. B. Konfigurationsdaten, die zu der Konfiguration des Prozesses 16 gehören. Jeder der Workstations 14 enthält ferner einen Prozessor 21, der die Anwendungen ausführt, um einen Benutzer in die Lage zu versetzen, Prozeßsteuerroutinen zu erstellen und diese Prozeßsteuerroutinen zu der Steuereinrichtung 12 herunterzuladen. Entsprechend enthält die Steuereinrichtung 12 einen Speicher 22 zum Speichern von Konfigurationsdaten und Prozeßsteuerroutinen, die zur Steuerung des Prozesses 16 zu verwenden sind, und enthält einen Prozessor 24, der die Prozeßsteuerroutinen ausführt, um eine Prozeßsteuerstrategie zu implementieren. Wenn die Steuereinrichtung 12 eine DeltaV-Steuereinrichtung ist, kann sie eine grafische Darstellung der Prozeßsteuerroutinen innerhalb der Steuereinrichtung 12 einem Benutzer über eine der Workstations 14 zur Verfügung stellen, welche die Steuerelemente innerhalb der Prozeßsteuerroutine und die Art und Weise, in der diese Steuerelemente konfiguriert sind, um die Steuerung des Prozesses 16 zu schaffen, dargestellt.

35 In dem in Fig. 1 gezeigten Prozeßsteuernetz 10 ist die Steuereinrichtung 12 über den Bus 18 mit drei Garnituren von ähnlich konfigurierten Reaktoren, die hierin als Reaktor 01, Reaktor 02 und Reaktor 03 bezeichnet werden, kommunikativ verbunden. Der Reaktor_01 enthält ein Reaktorgefäß 100, zwei Zulaufventile 101 und 102, die so angeschlossen sind, daß sie Fluideinlaßleitungen steuern, die Fluid in das Reaktorgefäß 100 abgeben, und ein Auslaßventil 103, das so angeschlossen ist, daß es einen Fluidfluß aus dem Reaktorgefäß 100 über eine Fluidauslaßleitung steuert. Eine Einrichtung 105, bei der es sich um einen Sensor, wie z. B. einen Temperatursensor, einen Drucksensor, eine Fluidpegelmessungseinrichtung oder eine andere Ausrüstung, wie z. B. eine elektrische Heizung oder eine Dampfheizung handeln kann, ist im oder nahe an dem Reaktorgefäß 100 angeordnet. In ähnlicher Weise enthält der Reaktor_02 ein Reaktorgefäß 200, zwei Zulaufventile 201 und 202, ein Auslaßventil 203 und eine Einrichtung 205, während der Reaktor_03 ein Reaktorgefäß 300, zwei Zulaufventile 301 und 302, ein Auslaßventil 303 und eine Einrichtung 305 enthält. Wie Fig. 1 zeigt, ist die Steuereinrichtung 12 mit den Ventilen 101-103, 201-203 und 301-303 und den Einrichtungen 105, 205 und 305 über den Bus 18 in Kommunikationsverbindung, um den Betrieb dieser Elemente zu steuern und einen oder mehrere Betriebsabläufe der Reaktor-Einheiten auszuführen. Derartige Betriebsabläufe können beispielsweise das Füllen der Reaktorgefäße, das Erwärmen des Materials innerhalb der Reaktorgefäße, das Leeren der Reaktorgefäße, das Reinigen der Reaktorgefäße, etc. einschließen.

50 Die Ventile, Sensoren und weiteren Geräte, die in Fig. 1 dargestellt sind, können jede gewünschte Art oder jeden Typ von Geräten einschließen, darunter beispielsweise Feldbus-Einrichtungen, Standard 4-20 mA-Einrichtungen, HART-Einrichtungen, etc. und können mit der Steuereinrichtung 12 unter Verwendung jedes bekannten oder gewünschten Kommunikationsprotokolls, wie z. B. des Feldbus-Protokolls, des HART-Protokolls, des 4-20-mA-Analogprotokolls etc. kommunizieren. Ferner können weitere Typen von Einrichtungen gemäß den Prinzipien der vorliegenden Erfindung mit der Steuereinrichtung 12 verbunden und von dieser gesteuert werden. Ferner können weitere Steuereinrichtungen mit der Steuereinrichtung 12 und mit den Workstations 14 über die Ethernet-Kommunikationsleitung 15 verbunden sein, um andere Einrichtungen oder Bereiche zu steuern, die zu dem Prozeß 16 gehören, und der Betriebsablauf derartiger zusätzlicher Steuereinrichtungen kann mit dem Betriebsablauf der in Fig. 1 gezeigten Steuereinrichtung 12 in jeder gewünschten Weise koordiniert werden.

60 Allgemein ausgedrückt kann das Prozeßsteuersystem von Fig. 1 verwendet werden, um Stapelprozesse umzusetzen, in welchen beispielsweise eine der Workstations 14 oder die Steuereinrichtung 12 eine Stapelausführungsroutine ausführen, bei der es sich um eine Steuerroutine höherer Ebene handelt, welche den Betrieb einer oder mehrerer Reaktor-Einheiten (sowie anderer Einrichtungen) leitet, um eine Reihe von unterschiedlichen Schritten (allgemein als Phasen bezeichnet), die zur Herstellung eines Produkts, wie z. B. eines Lebensmittelprodukts, eines Arzneimittels etc. erforderlich sind. Um unterschiedliche Phasen zu implementieren, verwendet die Stapelausführungsroutine ein allgemein so bezeichnetes Rezept, das die auszuführenden Schritte, die Mengen und Zeiten, die zu den Schritten gehören, und die Reihenfolge der Schritte festlegt. Schritte für ein Rezept können beispielsweise das Füllen eines Reaktorgefäßes mit den geeigneten Materialien oder Inhaltsstoffen, das Mischen der Materialien in dem Reaktorgefäß, das Erwärmen der Materialien in dem

Reaktorgefäß auf eine bestimmte Temperatur über eine bestimmte Zeitdauer, das Leeren des Reaktorgefäßes und anschließend das Reinigen des Reaktorgefäßes in Vorbereitung für den nächsten Stapelablauf einschließen. Jeder der Schritte definiert eine Phase des Stapelablaufes und die Stapelausführungsroutine innerhalb der Steuereinrichtung 12 führt für jede dieser Phasen einen unterschiedlichen Steueralgorithmus aus. Selbstverständlich können die bestimmten Materialien, Materialmengen, Erwärmungstemperaturen und Zeiten etc. für unterschiedliche Rezepte verschieden sein und folglich können diese Parameter von Stapelablauf zu Stapelablauf in Abhängigkeit von dem hergestellten Produkt und dem verwendeten Rezept geändert werden. Der Durchschnittsfachmann erkennt, daß hierin zwar Steuer Routinen und Konfigurationen für Stapelabläufe in den in Fig. 1 dargestellten Reaktor-Einheiten beschrieben werden, die Steuer Routinen jedoch zur Steuerung anderer gewünschter Einrichtungen zur Durchführung jedes anderen gewünschten Stapelprozeßablaufes oder zur Durchführung von kontinuierlichen Prozeßabläufen verwendet werden können, falls dies erwünscht ist.

Wie der Durchschnittsfachmann versteht, können dieselben Phasen oder Schritte eines Stapelprozesses an jeder der verschiedenen Reaktor-Einheiten in Fig. 1 gleichzeitig oder zu unterschiedlichen Zeiten implementiert werden. Da ferner die Reaktor-Einheiten in Fig. 1 allgemein dieselbe Anzahl und dieselbe Art von Geräten enthalten (das heißt sie gehören zur selben Einheitenklasse), kann dieselbe generische Phasensteuer routine für eine bestimmte Phase verwendet werden, um jede der verschiedenen Reaktor-Einheiten zu steuern, mit der Ausnahme, daß diese generische Phasensteuer routine modifiziert werden muß, um die verschiedene Hardware oder die unterschiedlichen Geräte, die zu den verschiedenen Reaktor-Einheiten gehören, zu steuern. Um beispielsweise eine Füllphase für den Reaktor 01 (während welcher die Reaktor-Einheit gefüllt wird) zu steuern, öffnet eine Füllsteuer routine ein oder mehrere Zulaufventile 101 oder 102 für eine bestimmte Zeitdauer, beispielsweise bis der Fluidpegelmesser 105 feststellt, daß das Gefäß 100 voll ist. Dieselbe Steuer routine kann jedoch verwendet werden, um eine Füllphase für den Reaktor 02 umzusetzen, indem nur die Bezeichnung der Zulaufventile so geändert wird, daß anstelle der Ventile 101 oder 102 die Ventile 201 oder 202 bezeichnet werden, und indem die Bezeichnung des Fluidpegelmessers so geändert wird, daß anstelle des Fluidpegelmessers 105 der Fluidpegelmesser 205 bezeichnet ist.

In bisher bekannten Systemen wäre eine generalisierte Steuer routine geschaffen worden, um eine bestimmte Phase in jeder der Reaktor-Einheiten (Reaktor 01, Reaktor 02 oder Reaktor 03) unter Verwendung von Aliasnamen (das heißt allgemeinen Variablen) auszuführen, um bestimmte, noch nicht spezifizierte Parameter, wie z. B. die jeweilige Reaktor-Einheit oder die jeweiligen Ventile oder Sensoren einer Reaktor-Einheit, an welchen ein Befehl der Routine anzuwenden war, anzugeben. Auf diese Weise wäre eine generalisierte Steuer routine (wie z. B. eine Füll routine) geschaffen worden, die während der Füllphase jeder der Reaktor-Einheiten in Fig. 1 zu verwenden wäre, und anstatt, daß bei der Erstellung der Steuer routine ein bestimmtes Ventil festgelegt worden wäre, das zum Füllen eines bestimmten Reaktorgefäßes zu öffnen ist, wurde die generalisierte Steuer routine so geschrieben, daß nur ein "Zulauf_Ventil" als ein Alias für das tatsächlich zu verwendende Ventil angegeben war. Für diese Systeme wäre eine Alias-Auflösungstabelle für jede der Reaktor-Einheiten erstellt worden, an welchen die generalisierte Steuer routine anzuwenden war. Die Alias-Auflösungstabelle für den Reaktor 01 könnte beispielsweise festlegen, daß das Alias "Zulauf_Ventil" das Ventil 101 ist, während die Alias-Auflösungstabelle für den Reaktor 02 festlegen könnte, daß das Alias "Zulauf_Ventil" das Ventil 201 ist, etc.

Wie vorstehend angegeben wurde bei einem System nach dem Stand der Technik das generalisierte Programm, das diese Aliasnamen verwendete, in einer Workstation erstellt und eine diesbezüglich konkretisierte Version dieses Programms wurde für jede Reaktor-Einheit (oder jedes andere Modul) unter Verwendung einer Alias-Auflösungstabelle für die bestimmte Reaktor-Einheit (oder das andere Modul) erstellt. Diese instantiierten Steuer Routinen wurden anschließend in die Steuereinrichtung heruntergeladen und darin gespeichert und anschließend während der Laufzeit verwendet, um die Phase in der bestimmten Einheit auszuführen, was großen Speicherplatz in der Steuereinrichtung erforderte. Bei einem anderen System nach dem Stand der Technik wurde die generalisierte Steuer routine in der Steuereinrichtung gespeichert und während der Laufzeit verwendet, um den programmierten Betriebsablauf oder die Phase an einer der Einheiten, an der sie angewendet wurden, durchzuführen. In diesem Fall wurden die Aliasnamen fliegend während der Laufzeit unter Verwendung der Alias-Auflösungstabelle für die bestimmte Einheit, die gesteuert wurde, aufgelöst. Bei dieser Konfiguration mußte dann, wenn an einer generalisierten Steuer routine, die gegenwärtig von der Steuereinrichtung abgearbeitet wurde, eine Veränderung vorzunehmen war, der Lauf dieser Routine unterbrochen werden, um es zu ermöglichen, eine neue generalisierte Steuer routine in die Steuereinrichtung herunterzuladen.

Ferner erlaubte es keines dieser bekannten Systeme, daß eine einzige generische Prozeßsteuer routine mit Aliasnamen an mehreren oder an unterschiedlichen Arten von Einheiten oder Hardwaresystemen angewandt wurde. Tatsächlich war bei diesen Systemen nach dem Stand der Technik eine Steuer routine für eine Phase auf die Verwendung mit einer Einheitenklasse, das heißt einer bestimmten Art von Hardwareeinheit zur Durchführung einer Art von Funktion innerhalb des Prozeßsteuernetzes beschränkt. Als Folge mußte eine erste generische Prozeßsteuer routine geschaffen und gespeichert werden, um Reaktor-Einheiten zu füllen, während eine unterschiedliche generische Prozeßsteuer routine geschaffen und gespeichert werden mußte, um Misch tanks zu füllen und wiederum eine unterschiedliche Routine zum Füllen von Zulauf tanks, was zu der Erstellung von vielen verschiedenen generischen Steuer Routinen zur Durchführung von im wesentlichen derselben Funktion an unterschiedlichen Arten von Hardwareeinheiten führte.

Ferner setzte wie vorstehend angegeben keines dieser Systeme nach dem Stand der Technik die generische Steuer routine in die Lage, kleinere Unterschiede zwischen den Geräten, die zu den unterschiedlichen Modulen einer bestimmten Art von Einheit oder Einheitenklassen gehörten, zu berücksichtigen. Im Gegenteil mußte eine unterschiedliche Phasenroutine geschrieben und an unterschiedlichen Modulen derselben Einheitenklasse verwendet werden, wenn diese unterschiedlichen Einheiten eine geringfügig verschiedene Hardware aufwiesen, wie z. B. eine elektrische Heizung in einem Fall oder eine Dampfheizung in einem anderen Fall. Dies machte es wiederum erforderlich, daß der Programmierer einen unterschiedlichen Prozeßsteuer algorithmus oder eine unterschiedliche Phasenklasse für unterschiedlichen Einheiten derselben Einheitenklasse verwendete, obgleich diese unterschiedlichen Einheiten im wesentlichen dieselbe Funktion innerhalb des Prozesses ausführten. Ferner hatten diese Systeme nach dem Stand der Technik kein einfaches Verfahren zum Spezifizieren eines Parameters, der während der Laufzeit der Prozeßsteuer routine zu identifizieren war, und wenn eine

derartige dynamische Referenz zulässig war, gab es keinen Weg, festzustellen, ob die dynamische Referenz einen gültigen Wert hatte, bevor versucht wurde, einen Befehl auf der Basis dieser Referenz auszuführen.

Die Vorgehensweise oder das System, das indirekte Referenzierung in Prozeßsteuerprogrammen oder -routinen wie hierin beschrieben verwendet, spricht einige der Probleme bei den vorstehend beschriebenen Systemen nach dem Stand der Technik an. Das hierin beschriebene System ermöglicht es, daß eine generische Prozeßsteuerroutine oder Phasensteuerroutine unter Verwendung von Aliasnamen geschrieben wird, daß diese jedoch weiterhin in der Steuereinrichtung unter Verwendung von kleinstem Speicherplatz gespeichert wird und in einer Weise ausgeführt wird, die es ermöglicht, daß die generische Phasensteuerroutine verändert wird, ohne daß veranlaßt wird, daß eine der gegenwärtig ablaufenden Routinen abgebrochen wird. Das hierin beschriebene System ermöglicht es ferner, daß eine Phasensteuerroutine für mehrere verschiedene Einheitenklassen oder unterschiedliche Hardwaretypen geschaffen und an diesen angewandt wird und verwendet eine Phasensteuerroutine, welche verifizierbare und leicht anwendbare dynamische Referenzen, das heißt Referenzen, die während der Laufzeit der Phasensteuerroutine eingebunden werden, enthalten kann.

Allgemein gesprochen basiert die Art, in der der Ablauf des Prozesses 16 in der Steuereinrichtung 12 verwaltet oder organisiert wird, auf der Wechselwirkung einer Anzahl von Objekten, von welchen jedes Attribut hat und zu dem eine oder mehrere Methoden gehören können. Jedes Objekt kann eine Anzahl von Subobjekten (oder Klassen) haben, die zu diesem gehören, wobei jedes Subobjekt Subobjekte haben kann und so fort. In einem generischen Sinn ist die Gesamtsteuerstrategie für den Prozeß 16 unter Verwendung eines objektorientierten Programmierparadigmas konfiguriert, was nach dem Stand der Technik bekannt ist und somit hier nicht im Detail beschrieben wird. Fig. 2 beschreibt ein Beispiel einer Objekthierarchie, welche die Beziehung zwischen einer Anzahl von Objekten darstellt, die zu dem Prozeßsteuer-
netz 10 in Fig. 1 gehören. Diese Hierarchie wird verwendet, um die Art zu erläutern, in der die Prozeßsteuerroutine auf beispielsweise einer der Workstations 14 erstellt wird und anschließend innerhalb der Steuereinrichtung 12 heruntergeladen und ausgeführt wird, und um den Kontext zu identifizieren, in dem diese Prozeßsteuerroutine arbeitet. Es versteht sich jedoch, daß die Art, in der die Prozeßsteuerroutinen erstellt und in der Steuereinrichtung 12 gespeichert werden, auf anderen Objekthierarchien basieren kann oder auf Objekthierarchien, die andere gewünschte Elemente oder Objekte enthalten.

In dem Objektbaum in Fig. 2 sind bestimmte Objekte mit Rahmen dargestellt, während allgemeine Kategorien von Objekten (oder Objekttypen) über den Objekten in dem Baum ohne Rahmen dargestellt sind. Wie Fig. 2 zeigt, enthält das Prozeßsteuernetz 10 ein oder mehrere Gebiete, die beispielsweise Gebäude oder andere geographische Bereichsbezeichnungen innerhalb einer Anlage sein können. In dem Objektbaum in Fig. 2 hat der Prozeß 16 drei Bereichsobjekte, die als Gebäude_01, Gebäude_02 und Gebäude_03 bezeichnet sind. Jedes Bereichsobjekt kann in Prozeßzellen eingeteilt sein, von welchen jede einen unterschiedlichen Aspekt des Prozesses kennzeichnet, der in dem Bereich durchgeführt wird. Das Bereichsobjekt Gebäude_01 in Fig. 2 ist so dargestellt, daß es zwei Prozeßzellenobjekte enthält, die mit Zelle_01 und Zelle_02 bezeichnet sind. Zelle_01 kann beispielsweise mit der Herstellung eines Bestandteils eines Produkts in Beziehung stehen, der in Zelle_02 verwendet wird. Jedes Zellenobjekt kann Null oder mehr Einheitenklassen enthalten, welche unterschiedliche Kategorien oder Gruppierungen von Hardware kennzeichnen, die in der Prozeßzelle verwendet werden. Allgemein ausgedrückt ist eine Einheitenklasse ein benanntes Objekt, das eine allgemeine Konfiguration einer Garnitur von parallel vorhandenen Geräten enthält, und ist insbesondere eine Sammlung von Einheiten, die eine sehr ähnliche, wenn nicht gar identische Prozeßinstrumentierung haben, von welchen jede eine sehr ähnliche, wenn nicht gar identische Funktion innerhalb eines Prozesses ausführt.

Einheitenklassenobjekte sind typischerweise so benannt, daß sie die Arten von Einheiten innerhalb des Prozeßsteuersystems beschreiben, zu dem sie gehören. Fig. 2 enthält eine Misch_Tank-Einheitenklasse, eine Reaktor-Einheitenklasse und eine Zuliefer_Tank-Einheitenklasse. Selbstverständlich sind in den meisten Prozeßsteuernetzen viele andere Typen von Einheitenklassen vorgesehen oder definiert, wie beispielsweise Trocknereinheiten, Gießtrichtereinheiten und andere einzelne oder logische Gruppierungen von Hardware enthalten.

Wie bei der Reaktor-Einheitenklasse in Fig. 2 dargestellt, kann jedes Einheitenklassenobjekt Einheitenmodulobjekte und Phasenklassenobjekte haben, die zu diesem gehören. Einheitenmodulobjekte bezeichnen allgemein bestimmte Instanzen von parallel vorhandener Hardware innerhalb der benannten Einheitenklasse, während Phasenklassen allgemein die Phasen festlegen, die an den Einheitenmodulen angewandt werden können, die zu der Einheitenklasse gehören. Genauer ausgedrückt ist ein Einheitenmodulobjekt ein benanntes Objekt, das alle Variablen und Einheitenphasen (weiter unten definiert) für eine einzelne Prozeßeinheit enthält, und ist typischerweise so benannt, daß eine bestimmte Prozeß-ausrüstung bezeichnet ist. Beispielsweise hat die Reaktor-Einheitenklasse in Fig. 2 Reaktor_01-, Reaktor_02- und Reaktor_03-Einheitenmodule, die Reaktor_01, Reaktor_02 und Reaktor_03, welche in Fig. 1 dargestellt sind, jeweils entsprechen. Die Misch_Tank-Einheitenklasse und die Zuliefer_Tank-Einheitenklasse haben in ähnlicher Weise bestimmte Einheitenmodule, die der jeweiligen Hardware oder der jeweiligen Ausrüstung in dem Prozeß 16 entsprechen. Der Einfachheit halber ist jedoch keine Ausrüstung, die zu der Misch_Tank- oder der Zuliefer_Tank-Einheitenklasse gehört, in Fig. 1 dargestellt.

Eine Phasenklasse ist ein benanntes Objekt, das die allgemeine Konfiguration für eine Phase enthält, die auf den mehreren Einheiten ablaufen kann, die zu derselben Einheitenklasse gehören, und gemäß vorliegender Erfindung auf mehreren unterschiedlichen Einheitenklassen. Im wesentlichen ist jede Phasenklasse eine unterschiedliche Steuerroutine (oder Phase), die durch die Steuereinrichtung 12 erstellt und verwendet wird, um die Einheitenmodule innerhalb derselben oder innerhalb von verschiedenen Einheitenklassen zu steuern. Typischerweise ist jede Phasenklasse in Übereinstimmung mit dem Verb benannt, das eine Aktion beschreibt, die an einem Einheitenmodul ausgeführt wird. Beispielsweise hat, wie in Fig. 2 dargestellt, die Reaktor-Einheitenklasse eine Füllphasenklasse, die verwendet wird, um eines der Reaktorgefäße 100, 200 oder 300 in Fig. 1 zu füllen, eine Heizphasenklasse, die verwendet wird, um eines der Reaktorgefäße 100, 200 oder 300 aus Fig. 1 aufzuheizen, eine Entleerungsphasenklasse, die verwendet wird, um eines der Reaktorgefäße 100, 200 oder 300 in Fig. 1 zu entleeren, und eine Reinigungsphasenklasse, die verwendet wird, um eines der Reaktorgefäße 100, 200 oder 300 aus Fig. 1 zu reinigen. Selbstverständlich können beliebige andere Phasenklassen zu dieser oder jeder anderen Einheitenklasse gehören. Es sei angemerkt, daß die Füllphasenklasse sowohl zu der Reaktor-

Einheitenklasse als auch der Zuliefer_Tank-Einheitenklasse gehört und so gemäß vorliegender Erfindung verwendet werden kann, um die Füllfunktion an den Reaktor-Einheitenmodulen sowie den Zuliefer_Tank-Einheitenmodulen zu vollziehen.

Eine Phasenklasse kann allgemein als eine Subroutine betrachtet werden, die durch die Stapelausföhrungsroutine aufzurufen ist, um eine in einem Gesamtstapelprozeß erforderliche Funktion gemäß der Definition durch das Rezept für den Stapelprozeß auszuführen. Eine Phasenklasse kann Null oder mehr Phaseneingabeparameter enthalten, welche im wesentlichen die Eingaben sind, die von der Stapelausföhrungsroutine oder einer anderen Phasenklasse an die Phasenklassen-Subroutine abgegeben werden, Null oder mehr Phasenausgabeparameter, welche im wesentlichen die Ausgaben der Phasenklassen-Subroutine sind, die zu der Stapelausföhrungsroutine oder einer anderen Phasenklasse zurückgeleitet werden, Null oder mehr Phasenmitteilungen, welche Mitteilungen sein können, die dem Benutzer bezüglich des Betriebsablaufes der Phasenklasse angezeigt werden können, Informationen in Bezug auf andere Phasenklassen, mit welchen diese Phasenklasse in irgendeiner Weise zusammengehört, und Null oder mehr Phasenalgorithmusparameter, welche veranlassen, daß in Phasenlogikmodulen (PLM) oder Einheitenphasen basierend auf dieser Phasenklasse Parameter geschaffen werden. Diese Phasenalgorithmusparameter werden als zeitweilige Speicherorte oder Variable während der Ausführung der Phase verwendet und sind nicht notwendigerweise für den Benutzer oder die Stapelausföhrungsroutine sichtbar. Wichtig ist, daß die Phasenklasse eine oder mehrere Phasenalgorithmusdefinitionen (PAD) enthält, welche allgemein ausgedrückt Steuerrouinen sind, die zum Implementieren der Phase verwendet werden. Ferner hat die Phasenklasse eine Liste von Zuordnungen zu Null, einer, zwei oder mehr Einheitenklassen und diese Liste definierte die Einheitenklasse, für welche diese Phasenklasse und folglich die PAD der Phasenklasse angewandt werden kann. Die Füllphasenklassenliste von Zuordnungen würde sowohl die Reaktor-Einheitenklasse als auch die Zuliefer_Tank-Einheitenklasse enthalten.

Die PAD ist ein unbenanntes Objekt, das die abstrakte oder generische Phasenlogik (Algorithmus) für eine Phasenklasse enthält und unter Verwendung jeder gewünschten Art von Programmierstruktur konfiguriert sein kann, wie z. B. einer impliziten Phasenlogik-Statusmaschine, sequentieller Flußdiagramm-Komposite (SFC), Funktionsblock-Komposite oder jeder anderen gewünschten Steuerprogrammierlogik, die durch die Steuereinrichtung 12 verwendet werden kann, um den Betriebsablauf des Prozesses 16 zu steuern. In einer Ausführungsform werden PADs unter Verwendung von SFC-Programmiertechniken definiert, in welchen eine Anzahl von Schritten zusammengebunden werden und die Aktionen innerhalb eines Schrittes ausgeführt werden, bis ein Übergangszustand wahr wird, und an diesem Punkt bewegt sich die Steuerung zu dem nächsten Schritt, der durchgeführt wird, bis der nächste Übergangszustand wahr wird, und so fort. Das SFC-Programmierprotokoll basiert auf den Normen IEC 848 und IEC 1131-3 für Programmiersprachen und ist nach dem Stand der Technik bekannt. Die PADs können jedoch jede andere gewünschte Art von Programmierstruktur verwenden, um den Betriebsablauf einer Phase zu definieren. Allgemein gesprochen schafft die PAD die grundsätzliche oder generische Steueroutine, die von der Steuereinrichtung 12 während des Betriebsablaufes eines Stapelprozesses auszuführen ist.

Um es zu erlauben, daß eine generische PAD auf eine beliebige Anzahl von unterschiedlichen Einheitenmodulen innerhalb einer Einheitenklasse angewandt wird, wird die PAD so konfiguriert (unter Verwendung einer Konfigurationsanwendung in der Workstation 14 durch den Benutzer), daß sie jedes externe Modul oder I/O referenziert, das von Einheitenmodul zu Einheitenmodul unterschiedlich ist, sowie jede andere gewünschte Variable oder jeden anderen gewünschten Parameter, und zwar unter Verwendung eines Aliasnamen, das heißt eines generischen oder noch nicht spezifizierten Namens, der noch nicht mit einer bestimmten Hardware oder einem bestimmten Hardwareelement verbunden ist. Als Resultat ist die PAD in dem Sinn generisch, daß die PAD auf mehrere Einheitenmodule und auch in mehreren Einheitenklassen angewandt oder verwendet werden kann.

Jede der Phasenklassen von Fig. 2 enthält ein PAD-Objekt, mit der Ausnahme der Heizphase, die zwei PAD-Objekte enthält, welche es ermöglichen, daß dieselbe Phasenklasse auf Steuereinheiten (wie z. B. Reaktor-Einheiten) angewandt und von diesen verwendet wird, die geringfügig unterschiedliche Arten von Hardware oder Ausrüstung aufweisen, wie nachfolgend im Detail beschrieben wird.

Wie anhand der Einheitenmodule in Fig. 2 gezeigt ist, enthält jedes Einheitenmodulobjekt Null oder mehr Einheitenkennzeichen (UT) oder Parameter, die Ausgangswerte haben. Diese Parameter können Einstellungen und Konfigurationsparametern der Geräte, die zu dem Einheitenmodul gehören, entsprechen. Ferner kann jedes Einheitenmodulobjekt Alarmsignale, Ressourcenidentifikationen, eine Kontrollanzeige (wie z. B. Mensch-Maschine-Schnittstellenbild), eine Liste der Ressourcen, die dieses Einheitenmodul benötigt, und Prozeßzelleninformationen aufweisen, die diesem zugeordnet sind. Dabei ist wichtig, daß jedes Moduleinheitenobjekt eine Alias-Auflösungstabelle (ART), Null oder mehr Einheitenphasenobjekte (UP) und eine Einheitenphasentabelle (UPT) aufweist, die zu diesem gehören. Die Alias-Auflösungstabelle ist ein unbenanntes Objekt, das eine Liste von Aliasnamen/Instanzendefinitionspaaren für das Einheitenmodul, zu welcher sie gehört, enthält. Die Liste der Aliasnamen in der Tabelle basiert auf den gültigen Aliasnamen, die für jede Einheitenklasse definiert sind, welche wiederum alle Aliasnamen einschließt, die in allen Phasenklassen verwendet werden, die zu der Einheitenklasse dieses Einheitenmoduls gehören. Mit anderen Worten enthält die Alias-Auflösungstabelle eines Einheitenmoduls eine Aliasdefinition für jedes Alias, das in jeder Phasenklasse verwendet wird, die an diesem Einheitenmodul implementiert werden kann. Der Benutzer konfiguriert die Instanzendefinition für Aliasnamen, die an jeder Einheit zur Zeit der Konfiguration basierend auf den Aliasnamen, die in den Phasenklassen verwendet werden, die zu der Einheitenklasse gehören, definiert werden sollten.

In einigen Instanzen ist es wünschenswert, einen speziellen Wert für eine Aliasinstanzendefinition zu schaffen, der bedeutet, einen bestimmten Aliasnamen zu IGNORIEREN. Der Effekt dieser Definition soll besagen, daß an diesem Einheitenmodul eine Phasenlogik, die diesen Aliasnamen verwendet, für das Herunterladen in die Steuereinrichtung 12 zugelassen werden sollte, obgleich dieses Einheitenmodul diesen Aliasnamen nicht braucht oder unterstützt. Die Ausführung während der Laufzeit verursacht, daß Schreibversuche in einen Parameter mit Alias der Phasenlogik, der mit einem Parameterpfadstring mit Alias spezifiziert wird und eine Aliasinstanzendefinition IGNORIEREN hat, unterdrückt werden, veranlaßt, daß Leseversuche aus einem Parameter mit Alias, der mit IGNORIEREN definiert ist, unterdrückt werden, oder sendet einen Nullwert oder einen anderen voreingestellten Wert zurück und verursacht möglicherweise, daß ein

Alarm ausgelöst wird, um einen Benutzer in Kenntnis zu setzen, daß ein Problem vorliegt. Falls erwünscht, kann die IGNORIEREN-Definition als ein Attribut der Aliasdefinition gespeichert werden und kann während der Laufzeit unter Verwendung beispielsweise einer IF...THEN-Logik getestet werden.

Jedes Einheitenphasenobjekt ist ein bekanntes Objekt, das eine Instanz einer Phasenklasse darstellt, die zu einem bestimmten Einheitenmodul gehört oder für ein solches geschaffen wurde. In dem Konfigurationssystem (das heißt in einer der Workstations 14) stellt das Einheitenphasenobjekt eine Komponente eines Einheitenmoduls dar, die unabhängig geändert und heruntergeladen werden kann. In dem Laufzeitsystem (das heißt in der Steuereinrichtung 12) stellt das Einheitenphasenobjekt eine Phasenlogik dar, die durch die Steuereinrichtung 12 an einem Einheitenmodul unabhängig betrieben (gestartet, gestoppt, gehalten, unterbrochen etc.) werden kann; möglicherweise parallel mit anderen Einheitenphasen, die gleichzeitig auf unterschiedlichen Einheitenmodulen aktiv sind. Im wesentlichen ist das Einheitenphasenobjekt die diesbezüglich konkretisierte Version einer der Phasenklassen, die unter Verwendung der Alias-Auflösungstabelle für das Einheitenmodul, zu welchem das Einheitenphasenobjekt gehört, aufgelöst wurde. Beispielsweise würde ein Einheitenphasenobjekt des Reaktors_02 in Fig. 1 unter Verwendung der generischen PAD für die Füllphasenklasse geschaffen, in welcher die Aliasnamen unter Verwendung der Alias-Auflösungstabelle für die Reaktor_02-Moduleinheit aufgelöst sind. Somit könnte das Alias Zulauf_Ventil in der PAD für die Füllphasenklasse als das Ventil 201 oder 202 in Fig. 1 in dem Füllphasenobjekt für das Reaktor_02-Einheitenmodul definiert werden. Die Steuereinrichtung 12 führt tatsächlich das Einheitenphasenobjekt aus (das heißt die diesbezüglich konkretisierte Version der Phasenklasse während der Laufzeit) und beläßt die generische Version der Phasenklasse in dem Speicher 22.

Die Phasentabelle für ein Einheitenmodul ist ein unbenanntes Objekt, das Schlüsseleigenschaften für jede Einheitenphase enthält, die an dem Einheitenmodul verfügbar gemacht wird. Die Phasentabelle enthält eine Liste von Phasenklassennamen aller Phasenklassen, die zu der Einheitenklasse des Einheitenmoduls gehören. Für jede Phasenklasse sieht bzw. konfiguriert der Benutzer die Schlüsseleigenschaften, einschließlich des Einheitenphasennamens (String), einer Verifizierungsangabe, daß eine gültige Aliasinstanzdefinition in der Alias-Auflösungstabelle des Einheitenmoduls für jeden Aliasnamen vorhanden ist, der in der Phasenklasse verwendet wird, und daß jede andere obligatorische Phasenklassenverifizierungsprüfung bestanden wurde, sowie eine Download-Angabe, die den Programmgestalter oder Benutzer in die Lage versetzt, das Herunterladen der Einheitenphasenlogik für Phasenklassen zu unterdrücken. Beispielsweise würde eine Einheitenphase nicht heruntergeladen, wenn entweder die Verifizierungsangabe festlegt, daß die Verifizierung nicht vorgekommen ist, oder wenn die Download-Angabe durch den Benutzer auf NEIN gesetzt wurde. Die Phasentabelle kann ferner eine permanente (Ja/Nein) Identifizierung enthalten, die die Steuereinrichtung 12 benachrichtigt, wenn diese Einheitenphase als "permanent" in dem Laufzeitsystem behandelt werden sollte. Wenn dies der Fall ist, wird die Einheitenphase stets instantiiert, so daß sie niemals aufgrund von Over-Commitment der Steuereinrichtungsspeicherressourcen nicht ablaufen kann. Andere Informationen können ebenfalls in der Einheitenphasentabelle vorgesehen sein, wie z. B. Eigenschaften, die von der Stapelausführungsroutine benötigt werden, Ressourcenidentifikationskennzeichen, benötigte Ressourcen, ein Ausführungsperioden-Teiler/Multiplikator für die Steuereinrichtung 12 und beliebige andere Eigenschaften, welche das Laufzeitverhalten einer Phase steuern würden.

Fig. 3 stellt eine detailliertere Version einiger der in Fig. 2 dargestellten Objekte dar und zeigt die Beziehungen zwischen diesen Objekten besser. Um die Prinzipien der vorliegenden Erfindung darzustellen, sind in Fig. 3 zwei Einheitenklassen dargestellt, nämlich eine Reaktor-Einheitenklasse 50 und eine Zuliefer_Tank-Einheitenklasse 52. Für die Reaktor-Einheitenklasse 50 ist ein Einheitenmodul 54 dargestellt, nämlich der Reaktor_01. Es können zwar auch andere vorhanden sein, die jedoch in Fig. 3 nicht dargestellt sind. Das Einheitenmodul 54 definiert einige der Reaktorparameter, die zu der Reaktor-Einheitenklasse gehören, nämlich daß die Kapazität des Reaktors_01 300 beträgt und daß der Reaktor_01 kein Rührwerk enthält. Entsprechend gehören zwei Phasenklassen zu der Reaktor-Einheitenklasse, darunter eine Füllphasenklasse 56 und eine Entleerungsphasenklasse 58. Die Füllphasenklasse 56 enthält eine PAD (als ein SFC in grafischer Form auf deren rechter Seite dargestellt), welche so gestaltet wurde, daß sie zwei Aliasnamen verwendet, nämlich #ZULAUF_VENTIL# UND #PEGEL#. Diese Aliasnamen werden tatsächlich in den in der PAD der Füllphasenklasse 56 dargestellten Rahmen verwendet, können jedoch alternativ irgendwo sonst innerhalb der Logik der PAD verwendet werden. Die Füllphasenklasse 56 enthält ferner eine Eingabe, die als SOIL_PEGEL definiert ist, und eine Ausgabe, die als END_PEGEL definiert ist. Während die Aliasnamen so angegeben sind, daß sie durch ein Nummernzeichen (#) begrenzt oder markiert sind, könnte jedes andere Identifizierungskennzeichen verwendet werden, um einen Aliasnamen zu definieren, der bei der Instantiierung einer Phase ersetzt werden muß. In ähnlicher Weise enthält die Entleerungsphasenklasse 58 eine PAD, die auf der rechten Seite derselben in grafischer Form dargestellt ist, welche die Aliasnamen #AUSLASS_VENTIL# und #PEGEL#, eine Eingabe, die als RATE definiert ist, eine Ausgabe, die als END_PEGEL definiert ist, und einen Algorithmusparameter (von der PAD verwendet) der als TATSÄCHLICHE_RATE definiert ist, hat, welche als eine vorübergehende Speicherposition während der Ausführung der PAD verwendet werden können.

Da die Füll- und Entleerungsphasenklassen 56 und 58 zu der Reaktor-Einheitenklasse 50 gehören, sollte das Reaktor_01-Einheitenmodul 54 (das ebenfalls zu der Reaktor-Einheitenklasse 50 gehört) so gestaltet sein, daß es die Aliasnamen unterstützt, die in den beiden Phasenklassen 56 und 58 verwendet werden, und zwar aus dem Grund, daß die Steuereinrichtung 12 versuchen kann, diesbezüglich konkretisierte Versionen der Phasenklassen 56 und 58 für das Reaktor_01-Einheitenmodul 54 während der Laufzeit zu schaffen. Als Resultat wird eine Alias-Auflösungstabelle 60 für das Reaktor_01-Einheitenmodul 54 geschaffen, um jeden der Aliasnamen #ZULAUF_VENTIL# (in der Füllphasenklasse 56 verwendet), #PEGEL# (sowohl in der Füllphasenklasse 56 als auch der Entleerungsphasenklasse 58 verwendet) und #AUSLASS_VENTIL# (in der Entleerungsphasenklasse 58 verwendet) zu definieren. Die Alias-Auflösungstabelle 60 enthält gültige Definitionen für #ZULAUF_VENTIL# und #PEGEL#-Aliase, enthält jedoch keine gültige Definition für das #AUSLASS_VENTIL#-Alias. Als Resultat bestimmt die von der Workstation 14 ausgeführte Konfigurationsroutine dann, wenn sie bestimmt, daß jede Phasenklasse innerhalb des Steuerschemas gültig definiert ist, daß ein Füllphasenobjekt für das Reaktor_01-Einheitenmodul 54 geschaffen werden kann, da jeder der Füllphasenklassen-Aliasnamen in der Alias-Auflösungstabelle 60 für das Reaktor_01-Einheitenmodul 54 gültig definiert ist. Diese Konfigurationsroutine bestimmt jedoch, daß ein Entleerungsphasenobjekt für das Reaktor_01-Einheitenmodul 54 nicht ge-

schaffen werden kann, da die Alias-Auflösungstabelle 60 keine gültige Definition für alle Aliasnamen aufweist, die von der Entleerungsphasenklasse 58 verwendet werden. Als Resultat zeigt eine Phasentabelle 62 für das Reaktor_01-Einheitenmodul 54 (welche durch die Konfigurationsanwendungen der Workstation 14 geschaffen wird) an, daß die Füllphase für das Reaktor_01-Einheitenmodul 54 aufgelöst ist und in die Steuereinheit 12 heruntergeladen werden kann, daß jedoch die Entleerungsphase für das Reaktor_01-Einheitenmodul 54 nicht aufgelöst ist und daher nicht in die Steuereinrichtung 12 heruntergeladen werden kann, und zwar bestimmt durch die ungültige Definition des #AUSLASS_VENTIL#-Alias in der Alias-Auflösungstabelle 60.

Es sei angemerkt, daß die in Fig. 3 gezeigte Füllphasenklasse 56 generisch ausreichend definiert ist, um auf eine zusätzliche Einheitenklasse anwendbar zu sein, nämlich die Zuliefer_Tank-Einheitenklasse 52, die in Fig. 3 so dargestellt ist, daß zu ihr ein Zuliefertank_06-Einheitenmodul 64 gehört. Als Resultat muß für die Füllphasenklasse, die zum Herunterladen in die Steuereinrichtung 12 definiert oder in ordnungsgemäßer Weise unterstützt wird, das Zuliefertank_06-Einheitenmodul (sowie alle anderen Einheitenmodule der Zuliefer_Tank-Einheitenklasse 52) eine Alias-Auflösungstabelle haben, die eine Definition für die von der Füllphasenklasse 56 verwendeten Aliase gibt, nämlich #ZULAUF_VENTIL# und #PEGEL#. Wenn dies erfüllt ist, kann die Füllphasenklasse 56 verwendet werden, um Einheitenphasen für jedes Einheitenmodul entweder in der Reaktor-Einheitenklasse 50 oder der Zuliefer_Tank-Einheitenklasse 52 zu erzeugen.

Während der Konfiguration verwendet ein Systemgestalter, wie z. B. ein Techniker, ein Konfigurationsprogramm, das auf einer der Workstations 14 ausgeführt wird, um die Phasenklassen und Alias-Auflösungstabellen für jedes der Einheitenmodule innerhalb des Prozeßsteuernetzes 10 zu konfigurieren. Allgemein ausgedrückt definiert der Techniker die PADs für jede der Phasenklassen (wie z. B. die Füll-, Heiz-, Entleerungs- und Reinigungsphasenklasse in Fig. 2) unter Verwendung jeder gewünschten Programmiersprache oder -umgebung und unter Verwendung von Aliasnamen für bestimmte Variable oder Module, wie z. B. Einrichtungen, Modulparameter, Funktionsblöcke, etc., so daß die PADs verwendet oder angewandt werden können, um jedes der Einheitenmodule zu steuern, das zu einer Einheitenklasse gehört, zu welcher die Phasenklasse gehört. Die Konfigurationsanwendung kann den Techniker in die Lage versetzen, diese PADs in jeder gewünschten Weise zu importieren und kann den Techniker auffordern, jede erforderliche Information anzugeben, darunter beispielsweise eine Liste von Aliasnamen, die in den PADs für jede Phasenklasse verwendet werden.

Gemäß vorliegender Erfindung kann der Techniker mehrere PADs für jede bestimmte Phasenklasse definieren, um Unterschiede bei der Hardware oder den Geräten, die zu den unterschiedlichen Einheitenmodulen derselben (oder verschiedener) Einheitenklassen gehören, zu berücksichtigen. Somit kann der Techniker beim Erstellen der Heizphasenklasse für die Reaktor-Einheitenklasse eine erste PAD schaffen, welche ein Reaktorgefäß unter Verwendung einer elektrischen Heizausrüstung erwärmt, und eine zweite PAD, welche ein Reaktorgefäß unter Verwendung einer Dampfheizausrüstung erwärmt. Jeweils eine unterschiedliche dieser PADs wird verwendet, um das Einheitenphasenobjekt für unterschiedliche Einheitenmodule zu schaffen. Wenn beispielsweise das Reaktor_01-Einheitenmodul mit einer elektrischen Heizausrüstung versehen ist (beispielsweise die Einrichtung 105 in Fig. 1 ein elektrisches Heizelement ist) und der Reaktor_02 mit einer Dampfheizausrüstung versehen ist (beispielsweise die Einrichtung 205 in Fig. 1 ein Dampfheizelement ist), wird die erste PAD der Heizphasenklasse verwendet, um die Heizeinheitenphase für das Reaktor_01-Einheitenmodul zu erstellen, während die zweite PAD der Heizphasenklasse verwendet wird, um die Heizeinheitenphase für das Reaktor_02-Einheitenmodul zu schaffen. Um die Workstation 14 und die Steuereinrichtung 12 in die Lage zu versetzen, zu entscheiden, welche PAD einer Phasenklasse zu verwenden ist, wenn eine bestimmte Einheitenphasenklasse für eine Phasenklasse geschaffen wird, die mehrere PADs hat, enthält jedes Einheitenmodul, das eine Phasenklasse mit mehreren PADs unterstützt, eine Angabe, welche den Gerätetyp identifiziert, den dieses Einheitenmodul hinsichtlich der Unterscheidung zwischen den unterschiedlichen PADs hat. Dieses Kennzeichen kann in jeder gewünschten Weise gespeichert werden, so lang es für die Steuereinrichtung 12 verfügbar ist, wenn die Steuereinrichtung 12 eine Einheitenphase zur Ausführung erstellt. Beispielsweise hat das Reaktor_01-Einheitenmodul einen Identifizierungsparameter, der auf einen Wert gesetzt ist, der angibt, daß dieser Reaktor eine elektrische Heizung hat, während das Reaktor_02-Einheitenmodul denselben Identifikationsparameter hat, der auf einen Wert gesetzt ist, der angibt, daß dieser Reaktor eine Dampfheizung hat. Wenn die Steuereinrichtung 12 eine Einheitenphase für ein Einheitenmodul schafft, greift sie auf den Identifikationsparameter zu und verwendet basierend auf dem Wert des Identifikationsparameters die erste PAD (elektrische Heizung) oder die zweite PAD (Dampfheizung), wenn die Einheitenphase erstellt wird.

Anschließend erstellt der Techniker die Alias-Auflösungstabelle für jedes Einheitenmodul (wie z. B. die Alias-Auflösungstabelle 60 in Fig. 3) und sieht in jeder Alias-Auflösungstabelle eine Definition für jeden der Aliasnamen vor, die in jeder einzelnen Phasenklasse verwendet werden, die zu der Einheitenklasse gehören, zu welcher das Einheitenmodul gehört. Beispielsweise würde in der Objekthierarchie in Fig. 2 der Techniker eine Alias-Auflösungstabelle für jedes der Einheitenmodule Reaktor_01, Reaktor_02 und Reaktor_03 erstellen, die eine Definition für jeden der Aliasnamen enthält, die in der Füll-, Heiz-, Entleerungs- und Reinigungsphasenklasse verwendet werden. Da die Füllphasenklasse 56 auch zu der Zuliefer_Tank-Einheitenklasse 52 gehört, wie in Fig. 3 am besten zu erkennen ist, müßte der Techniker sicherstellen, daß die Alias-Auflösungstabelle für jedes Einheitenmodul, das zu der Zuliefer_Tank-Einheitenklasse gehört (wie z. B. das Zuliefer_Tank06-Einheitenmodul 64 in Fig. 3), Definitionen für jeden der Aliasnamen enthält, die in der Füllphasenklasse 56 sowie in jeder anderen Phasenklasse verwendet werden, die zu der Zuliefer_Tank-Einheitenklasse 52 gehört. Selbstverständlich können, wie vorstehend angemerkt, einige der Aliasnamen mit dem IGNORIEREN-Wert in einigen der Alias-Auflösungstabellen der Einheitenmodule definiert sein, da sie für den Betriebsablauf dieses bestimmten Einheitenmoduls irrelevant sind.

Vorzugsweise enthält die Konfigurationsanwendung in der Workstation 14 eine Aliasdefinitionsprüfroutine, die automatisch prüft, ob jeder Aliasname für jede Phasenklasse von der Alias-Auflösungstabelle für jedes Einheitenmodul unterstützt wird, das zu der Einheitenklasse gehört, zu welcher die Phasenklasse gehört. In einer Ausführungsform ergibt jede Einheitenklasse eine Liste von Aliasnamen, das jeden Aliasnamen enthält, der in jeder Phasenklasse verwendet wird, die zu dieser Einheitenklasse gehört. Die Prüfroutine kann anschließend bestimmen, ob eine gültige Aliasdefinition für jeden dieser Aliasnamen in jeder Alias-Auflösungstabelle existiert, die zu dieser Einheitenklasse gehört. Da mehrere Einheitenklassen eine Phasenklasse teilen können (wie für die Füllphasenklasse in Fig. 2 und 3 dargestellt), können die-

selben Aliasnamen in unterschiedlichen Einheitenklassen verwendet werden, was es erfordert, daß Aliasnamen innerhalb des Systems global einzigartig sind. In einer weiteren Ausführungsform kann die Prüfroutine die Aliasnamen für eine bestimmte Phasenklasse bestimmen, die Einheitenklassen bestimmen, zu welchen die Phasenklasse gehört, und zwar basierend auf dem Einheitenklassenkennzeichen der Phasenklasse, und die Alias-Auflösungstabelle für jedes Einheitenmodul prüfen, das zu den identifizierten Einheitenklassen gehört, um zu bestimmen, ob diese Alias-Auflösungstabellen eine gültige Definition für jeden der Aliasnamen der Phasenklasse enthalten. Die Routine kann anschließend zu der nächsten Phasenklasse weitergehen und den Vorgang wiederholen, bis alle Phasenklassen geprüft wurden. Während dieses Prozesses kann die Prüfroutine die Phasentabelle für jedes Einheitenmodul ausfüllen und in der Phasentabelle angeben, ob jede Phase auflösbar ist, und zwar basierend auf der Alias-Auflösungstabelle für dieses Einheitenmodul, und ob diese Phase, das heißt Phasenklasse, in die Steuereinrichtung 12 zur Verwendung beim Ablaufbetrieb heruntergeladen werden kann. Die Prüfroutine kann ferner bestimmen, ob eine Definition für jeden Aliasnamen in einer bestimmten Alias-Auflösungstabelle vorhanden ist und ob die angegebene Definition eine gültige ist, das heißt eine gültige Position oder Einrichtung innerhalb des Prozeßsteuersystems angibt. Diese Überprüfung wird erreicht, indem eine Konfigurationsdatenbank innerhalb der Workstation 14 verwendet wird, welche so eingerichtet wird, daß sie die Systemhardwarekonfiguration und die von der Steuereinrichtung 12 während der Laufzeit verwendete Datenbank spiegelt. Die Verwendung dieser Prüfroutine hilft dabei, es zu ermöglichen, daß eine Phasenklasse durch mehrere Einheitenklassen unterstützt wird.

Die Phasentabellen können von dem Techniker verwendet werden, um zu bestimmen, welche Phasenklassen nicht von jedem einzelnen Einheitenmodul unterstützt werden, an welchen die Phasenklassen während der Laufzeit angewandt werden können. Wenn eine Phasenklasse auch nur von einem Einheitenmodul nicht unterstützt wird (bedingt durch eine ungültige oder fehlende Aliasdefinition in der Alias-Auflösungstabelle für dieses Einheitenmodul), verhindert die Konfigurationsroutine vorzugsweise, daß diese Phasenklasse in die Steuereinrichtung 12 heruntergeladen wird, um zu verhindern, daß die Steuereinrichtung 12 versucht, basierend der Phasenklasse eine ausführbare Routine zu schaffen, welche aufgrund der nicht auflösbaren Aliasdefinition unterbrochen oder angehalten werden kann. Ferner kann die Konfigurationsroutine das Herunterladen einer Phasenklasse basierend auf der Einstellung der Herunterladeparameter innerhalb der Phasentabelle verhindern.

Wenn alle Phasenklassen und Alias-Auflösungstabellen ordnungsgemäß konfiguriert sind, werden sie in die Steuereinrichtung 12 heruntergeladen, um den Ablaufbetrieb basierend auf diesen Objekten zu erlauben. Allgemein ausgedrückt speichert die Steuereinrichtung 12 die Alias-Auflösungstabellen und die Phasenklassen, die die Aliasnamen enthalten, in den Speicher 22. Die Phasenklassen und die Alias-Auflösungstabellen können jedoch in dem Speicher 20 einer der Workstations 14 oder in jedem anderen Speicher gespeichert werden, wenn dies erwünscht ist. In jedem Fall schafft die Steuereinrichtung 12 keine ausführbare Einheitenphasenroutine, bevor eine derartige Routine tatsächlich benötigt wird oder von der Stapelausführungsroutine aufgerufen wird (welche in einer Workstation 14 oder der Steuereinrichtung 12 gespeichert sein kann und auf diesen ablaufen kann). Wenn die Stapelausführungsroutine einen Stapelablauf implementiert, schafft sie zunächst eine Instantiierung jeder Phasenklasse für jedes der einzelnen Einheitenmodule, auf welchen der Stapelprozeß ablaufen soll. Die Steuereinrichtung 12 (oder ein Programm in dieser) greift auf die Phasenklassen, die zu verwenden sind, zu und greift auf die Alias-Auflösungstabelle für das Einheitenmodul zu, auf welchem die Phase, die zu der Phasenklasse gehört, ablaufen soll. Unter Verwendung der Alias-Auflösungstabelle und der PAD für die Phasenklasse erstellt die Steuereinrichtung 12 eine ausführbare Einheitenphase, wobei die Aliasnamen in der PAD aufgelöst werden oder durch die Definition für diese Namen innerhalb der Alias-Auflösungstabelle ersetzt werden. Wenn die Phasenklasse mehr als eine PAD hat, verwendet die Steuereinrichtung 12 die PAD-Identifizierungsparameter des Einheitenmoduls, um zu bestimmen, welche PAD zu verwenden ist, um die Einheitenphase zu schaffen. Anschließend führt die Steuereinrichtung 12 die Einheitenphase entsprechend der Anweisung von der Stapelausführungsroutine durch (das heißt die diesbezüglich konkretisierte Version der Phasenklasse).

Weil die Steuereinrichtung 12 die Phasenklasse, welche die Aliasnamen enthält, in ihrem Speicher 22 speichert, muß die Steuereinrichtung 12 keine diesbezüglich konkretisierte ausführbare Version jeder Phasenklasse für jedes Einheitenmodul (das heißt alle Einheitenphasen) zu jeder Zeit haben, was die Speicheranfordernisse der Steuereinrichtung 12 verringert. Tatsächlich muß gemäß vortiegender Erfindung die Steuereinrichtung 12 nur ausreichend Speicher verwenden, um jede der Phasenklassen und jede der instantiierten Einheitenphasen zu speichern, die gegenwärtig ablaufen. Nach der Ausführung einer Einheitenphase kann die Steuereinrichtung 12 die Einheitenphase entfernen, da die Steuereinrichtung 12 eine neue Einheitenphase für ein Einheitenmodul aus der gespeicherten Phasenklasse und der gespeicherten Alias-Auflösungstabelle für dieses Einheitenmodul erstellen kann. Selbstverständlich wird dann, wenn eine Einheitenphase für den Betriebsablauf der Steuereinrichtung 12 permanent instantiiert werden soll, wie von dem Benutzer in der Phasentabelle definiert, die Einheitenphase nicht entfernt, sondern in dem Steuereinrichtungsspeicher 22 gehalten, um sicherzustellen, daß für diese Einheitenphase stets Speicher verfügbar ist. In jedem Fall reduziert das Speichern der generischen, mit Alias versehenen Steuerrouinen (Phasenklassen), bis sie tatsächlich gebraucht werden, und das anschließende Erstellen einer ausführbaren Steueroutine unter Verwendung der Alias-Auflösungstabelle die erforderliche Speichergröße gegenüber den Systemen nach dem Stand der Technik, bei welchen es erforderlich ist, daß eine Steuereinrichtung ein separates ausführbares Programm für jede Phasenklasse für jedes Einheitenmodul zu jeder Zeit speichert. Da jedoch eine separate ausführbare Steueroutine (Einheitenphase) vor der Laufzeit geschaffen wird, erkennt die Steuereinrichtung 12, daß ein Auflösungsproblem zwischen der generischen Steueroutine und der Alias-Auflösungstabelle vorliegt, bevor sie mit der Ausführung eines Stapelablaufs beginnt, was verhindert, daß der Stapelablauf begonnen wird und anschließend während seines Betriebes unterbrochen wird, bedingt durch einen nicht auflösbaren Aliasnamen, was bei dem System nach dem Stand der Technik ein Problem war, bei welchem eine generische Steueroutine gespeichert und ausgeführt wurde, in welcher Aliasnamen fliegend während der Laufzeit aufgelöst wurden.

Da ferner eine ausführbare Einheitenphase vor der Laufzeit geschaffen wird und da diese Einheitenphase von der Steuereinrichtung 12 während der Laufzeit verwendet wird, ist die generische Phasenklasse stets verfügbar und somit kann diese Phasenklasse verwendet werden, um andere Einheitenphasen zu schaffen, während eine daraus geschaffene Einheitenphase abläuft. Entsprechend kann die generische Phasenklasse erweitert oder verändert werden, während eine aus

dieser Phasenklasse entwickelte Einheitenphase abläuft, was bedeutet, daß der Benutzer eine neue Phasenklasse herunterladen kann, ohne daß gegenwärtig in Ausführung befindliche Routinen, die aus der früheren Phasenklasse entwickelt wurden, unterbrochen werden müssen. Dies ist vorteilhaft, da es die Erweiterung der Steuereinrichtung 12 ohne Unterbrechung des gegenwärtig ablaufenden Prozesses erlaubt.

Da darüber hinaus eine Phasenklasse mit mehr als einer Einheitenklasse verbunden sein kann, kann eine einzelne Phasenklasse gespeichert werden und für mehrere unterschiedliche Typen von Einheiten oder Hardware verwendet werden, was die Anzahl der innerhalb des Steuersystems erforderlichen Objekte weiter reduziert und die Speicheranfordernisse der Steuereinrichtung 12 verringert. Da ferner eine Phasenklasse mehrere PADs haben kann, welche mit unterschiedlichen Geräten innerhalb derselben (oder verschiedenen) Einheitenklassen verwendet werden können, muß der Benutzer die Stapelausföhrungsroutine nicht so programmieren, daß die kleineren Unterschiede der Hardware berücksichtigt werden. Anstelle dessen berücksichtigt die Phasenklasse diese Unterschiede und somit kann die Stapelausföhrungsroutine dieselbe Phasenklasse verwenden oder aufrufen, um dieselbe Funktion an unterschiedlichen Einheitenmodulen durchzuführen, wenngleich auch die unterschiedlichen Einheitenmodule mit geringfügig unterschiedlicher Hardware ausgerüstet sind.

Die von dem Techniker entwickelten Prozeßsteuererroutinen können ferner bestimmte Parameter oder Variable enthalten, deren Wert angegeben werden kann, nachdem die diesbezüglich konkretisierte Prozeßsteuererroutine (das heißt die Einheitenphase) innerhalb der Steuereinrichtung 12 geschaffen wurde. Diese dynamisch gebundenen oder dynamisch aufgelösten Parameter sind nützlich, wenn beispielsweise einem Benutzer oder für die Stapelausföhrungsroutine innerhalb einer Phase eines Stapelablaufs auf einem bestimmten Einheitenmodul verschiedene Möglichkeiten zur Auswahl stehen. Beispielsweise kann eine Stapelausföhrungsroutine in Abhängigkeit von dem verwendeten Rezept entscheiden müssen, ob das Ventil 201 oder das Ventil 202 in dem Reaktor_02 in Fig. 1 geöffnet werden. Wenn beispielsweise der Stapelablauf die Herstellung eines mit Kohlensäure versetzten Getränks, wie z. B. Bier, betrifft, kann das Rezept die Herstellung von normalem Bier betreffen, in welchem Fall das Ventil 201 des Reaktors_02 während der Füllphase des Stapelprozesses geöffnet werden muß, oder das Rezept kann die Herstellung von Leichtbier betreffen, in welchem Fall das Ventil 202 des Reaktors_02 während der Füllphase des Stapelprozesses geöffnet werden muß. Anstatt daß zwei unterschiedliche Phasenklassen für diese unterschiedlichen Füllvorgänge (basierend auf dem Rezept) vorhanden sind, ist es nützlich, die Stapelausföhrung in die Lage zu versetzen, den Zulaufventilparameter dynamisch anzugeben, nachdem die Einheitenphase für das Reaktor_02-Einheitenmodul innerhalb der Steuereinrichtung 12 geschaffen wurde.

Wie vorstehend angeführt verwendeten Systeme nach dem Stand der Technik, welche dynamische Bindung ermöglichen, typischerweise ein Adressarray, in welchem unterschiedliche Zeiger an verschiedenen Adressen gespeichert werden konnten, welche innerhalb der Steuererroutine verwendet wurden, wobei zu jeder Adresse ein Zeiger gehörte. Es war jedoch schwierig, diese Adressen und die darin enthaltenen Zeiger zu verwalten und es gab keine Möglichkeit, dynamisch zu bestimmen, ob der Zeiger an einer Adresse gültig war, bevor versucht wurde, die dynamische Verbindung herzustellen. Wenn der Zeiger ungültig war, würde die Steuererroutine typischerweise unterbrochen, was insbesondere im Verlauf eines Stapelablaufs besonders unerwünscht war, da die Unterbrechung allgemein zum Verlust von Produktionszeit und -materialien führte und möglicherweise einen sehr schwierigen Eingriff der Bedienungspersonen erforderte, um den Stapelablauf fortzuführen.

Gemäß vorliegender Erfindung ist es bevorzugt, die Verwendung einer dynamisch gebundenen (dynamisch spezifizierten) Variablen oder eines entsprechenden Parameters (eines dynamischen Referenzparameters) in der Prozeßsteuererroutine für jede Phasenklasse zu ermöglichen. Mit anderen Worten ist es in einigen Fällen bevorzugt, einen dynamischen Referenzparameter in die PAD einer Phasenklasse zu setzen, wobei dieser dynamische Referenzparameter in die Einheitenphasen übertragen oder umgesetzt wird, wenn die ausführbare Einheitenphase aus der Phasenklasse geschaffen wird, wobei der Wert dieses dynamischen Referenzparameters spezifiziert werden kann, nachdem die Einheitenphase geschaffen wurde und auch nachdem die Ausführung der Einheitenphase begonnen hat (das heißt während der Laufzeit). Wie vorstehend angemerkt ist dieser Parameter nützlich, wenn beispielsweise die Auswahlentscheidung hinsichtlich des Wertes des Parameters auf einer Information basiert, die zum Zeitpunkt der Konfiguration nicht zur Verfügung steht, beispielsweise eine Auswahl, die auf einer Eingabe einer Bedienungsperson basiert, eine Auswahl, die auf einem Rezeptparameter basiert, der von der Stapelausföhrungsroutine weitergegeben wird, eine Auswahl, die auf Laufzeitwerten von Steuervariablen basiert, etc.

Der dynamische Referenzparameter zur Verwendung in der Prozeßsteuererroutine kann unter Verwendung von Konventionen definiert werden, die verwendet werden, um andere Hardware- oder Softwareparameter innerhalb des Prozeßsteuersystems zu definieren, mit der Ausnahme, daß der dynamische Referenzparameter in dem Speicher 22 der Steuereinrichtung 12 lokalisiert ist (oder einem anderen Speicher, falls erwünscht). Der dynamische Referenzparameter enthält vorzugsweise mehrere Attribute oder Felder, um die Ausführung von verschiedenen Operationen unter Verwendung des dynamischen Referenzparameters zu ermöglichen. Insbesondere kann der dynamische Referenzparameter die Felder oder Attribute enthalten, die in der folgenden Tabelle angegeben sind.

DE 100 11 661 A 1

Name	Typ	Konfigu- rierbar	Lesbar	Schreibbar
DREF	String	Nein	Lesen als String gibt den gegenwärtigen Referenzpfad wenn möglich mit aufgelösten Aliasnamen zurück. Nicht als Fließkomma, ganze Zahl, etc. lesbar.	Schreiben als String schafft einen neuen Referenzpfad; veranlaßt idealerweise die Verbindung einer neuen externen Referenz. Nicht schreibbar als Fließkomma, ganze Zahl, etc.
CONN	ganze Zahl	Nein	Definiert den Verbindungsstatu s der Referenz in dem DREF- Feld. Lesen als ganze Zahl: 0 bedeutet Referenz wurde hergestellt; -1 bedeutet Referenz wird niemals	Nein

			funktionieren; >0 bedeutet Verbindung im Aufbau, erneute Überprüfung später empfohlen.		5
DRFV	Fließ	Nein	Der Wert des referenzierten Feldes als ein Fließkommawert interpretiert. Wird in numerischen Ausdrücken verwendet. Wenn das referenzierte Feld nicht als ein Fließkommawert interpretiert werden kann (angenommen es ist ein Feld des String- Typs), ist der gelesene Wert MAX_Wert. Der Wert wird auch als MAX_Wert gelesen, wenn das CONN- Attribut nicht gleich 0 ist.	Schreiben in dieses Attribut verursacht, daß der Wert des referenzierten Feldes als ein Fließkommawert asynchron geschrieben wird. Wird bei numerischen Ausdrücken verwendet. Wenn das referenzierte Feld nicht als ein Fließkommawert interpretiert werden kann (ange- nommen es ist ein Feld des String- Typs), ist die Schreiboperation nicht durchführbar. Kein Schreibversuch, wenn das CONN- Attribut nicht gleich 0 ist.	10 15 20 25 30 35 40
DRSV	String	Nein	Der Wert des referenzierten Feldes als String interpretiert. In String- Ausdrücken verwendet. Wenn das referenzierte Feld numerisch ist, wird der numerische Wert	Schreiben in dieses Attribut verursacht, daß der Wert des referenzierten Feldes als String asynchron geschrieben wird. In String- Ausdrücken verwendet. Wenn das referenzierte Feld numerisch	45 50 55 60

			in String-Form umgewandelt (sofern möglich). Ein leerer String wird zurückgegeben, wenn das CONN- Attribut nicht gleich 0 ist.	ist, wird der String in die numerische Form umgewandelt (sofern möglich) und geschrieben. Kein Schreibversuch, wenn das CONN- Attribut nicht gleich 0 ist.
DRST	ganze Zahl	Nein	Liest den Wert des Statusfeldes des referenzierten Parameters (sofern vorhanden). Dieses Attribut wird verwendet, um den Statuswert von dem dynamisch referenzierten Feld zu einem lokalen Parameterstatusf eld zur Verwendung in nachfolgenden statussensitiven Berechnungen zu kopieren. Auf den MAX_Wert gesetzt, wenn das CONN- Attribut nicht gleich 0 ist; auf einen "gut"- Statuswert gesetzt, wenn das dynamisch referenzierte Feld kein Statusfeld hat.	Nein
WRST	ganze Zahl	Nein	Liest den Statuswert des referenzierten	Nein

			<p>Feldes und gibt den Erfolg des letzten Schreibvorganges in das dynamisch referenzierte Feld an. Werte könnte beliebige sein, die beispielsweise unterscheiden: "in Ausführung", "erfolgreich", "keine Kommunikation", "schlecht", etc.. Dieses Attribut wird auf einen Wert "schlecht" rückgesetzt, wenn ein neuer Referenz-String in das DREF-Attribut geschrieben wird.</p>	
--	--	--	--	--

Der Wert des DREF-Attributs (dynamische Referenz) ist der des Zeigers, des Kennzeichens oder des Pfades zu dem Parameter oder Feld, mit dem die dynamische Referenz gegenwärtig zugeordnet ist. In dem DeltaV-System ist dieser Zeiger ein String-Wert (wie z. B. ein Pfad), der auf ein Modul verweist, wie z. B. eine Einrichtung oder einen Modulparameter.

Beispielsweise könnte ein dynamischer Referenzparameter "AUSLASS_POS" einer String-Konstanten zugewiesen werden, so daß sie eine Referenz auf einen Parameter/ein Feld ist, beispielsweise unter Verwendung des Befehls

"AUSLASS_POS.DREF" = "VLV1004/AO/OUT.CV";

oder

"AUSLASS_POS.DREF" = "";//leere String-Konstante.

Selbstverständlich kann bei anderen Systemen der Wert des DREF-Feldes numerisch oder ein String sein, in Abhängigkeit von der Art, in der das System Geräte, Module, etc. spezifiziert. Allgemein ausgedrückt schafft die Verwendung einer DREF-Zuordnung einen neuen dynamischen Referenzparameter in der Steuereinrichtung 12, da die dynamischen Referenzen durch Zuordnen eines vollen Parameter-Referenzpfadstrings zu dem DREF-Feld eines dynamischen Referenzparameters geschaffen werden. Wenn eine dynamische Zielbestimmung von Steueroperationen erforderlich ist, muß der Benutzer einen dynamischen Referenzparameter für jeden Parameter schaffen, der nicht vor der Laufzeit bestimmt werden kann. Ein dynamischer Referenzparameter kann in jedem Kontext geschaffen werden, in dem jeder andere externe Referenzparameter in dem System geschaffen wird, einschließlich beispielsweise als eine Einheitenkennzeichenklasse, als Teil einer Einheitenklasse, als ein Phasenalgorithmusparameter in einer Phasenklasse und als ein Modullevelparameter in einem Phasenlogikmodul. Selbstverständlich kann der dynamische Referenzparameter in anderen Kontexten ebenso verwendet werden. Wie auch bei anderen benutzererstellten Parametern konfiguriert der Benutzer den Namen für den dynamischen Referenzparameter in der Weise, die den Parameternamenkonfigurationsregeln entspricht und die in dem lokalen Namensbereich einzigartig ist (das heißt er kann nicht den selben Namen wie andere Parameter, Blöcke oder Phasenklassennamen haben, die innerhalb derselben Ebene definiert sind). Das Schreiben eines String zu dem DREF-Attribut ist wahrscheinlich eine "teure" Operation (die alle früheren externen Referenzobjekte zerstört und eine Bindung an einen neuen externen Parameter vollzieht) und sollte demnach allgemein nur in nicht wiederholten (Impuls) Ausdrücken ausgeführt werden. Das Schreiben eines neuen String in dem DREF-Attribut sollte unmittelbar veranlassen, daß die aus dem CONN-Attribut gelesenen Werte (und den anderen Attributen des dynamischen Referenzparameters) "schlecht" werden (falls die Verbindung nicht unmittelbar geschaffen wird), so daß ein nachfolgender Ausdruck, der den Wert dieser Felder testet, zuverlässig arbeitet.

Die String-Zuordnung zu dem DREF-Attribut kann erreicht werden, indem jede gewünschte String-Operation einschließlich beispielsweise einer Verbindungsoperation (eine Verbindung von zwei oder mehr Strings), eine String-Auswahloperation (in der einer einer Vielzahl von möglichen Strings basierend auf dem Wert eines Operanden ausgewählt wird) oder jede andere String-Operation verwendet werden.

- Das CONN-Attribut (Verbindung) schafft eine Angabe, ob der durch das DREF-Attribut angegebene Wert ein gültiges oder auflösbares Feld innerhalb des Kontexts des Steuersystems ist. Wenn das DREF-Attribut verändert oder gesetzt wird, testet die Steuereinrichtung 12 unmittelbar und automatisch dessen Wert, um zu sehen, ob das Kennzeichen oder der Pfad für dieses existieren und in der gegenwärtigen Konfiguration des Steuersystems 10 lokalisiert oder aufgelöst werden können. Wenn der Wert des DREF-Attributs gültig und auflösbar ist, wird das CONN-Attribut auf 0 gesetzt.
- Wenn jedoch der Wert des DREF-Attributs nicht auflösbar ist und niemals aufgelöst werden kann, da er beispielsweise innerhalb des Kontexts des Steuersystems nicht ordnungsgemäß ist oder einfach nicht existiert, wird das CONN-Attribut auf -1 gesetzt. Wenn die Steuereinrichtung 12 aktiv versucht, das DREF-Attribut aufzulösen, jedoch aufgrund von Wartezeiten bedingt durch Verbindungsoperationen etc. nicht in der Lage war, dies zu vollziehen, wird der Wert des CONN-Attributs auf größer als 0 gesetzt, was angibt, daß dieser Wert noch nicht aufgelöst ist, jedoch später aufgelöst werden kann. Es ist bevorzugt, daß nach einer Zeitüberschreitungsperiode der Wert des CONN-Attributs auf -1 gesetzt wird, wenn das DREF-Attribut noch nicht aufgelöst wurde. Das CONN-Attribut ist sehr nützlich, da es das Testen der dynamischen Referenz während der Laufzeit ermöglicht. Beispielsweise kann ein einfacher Befehl "IF<Name des dynamischen Parameters>.CONN = 0, THEN <zu vollziehende Aktion>" verwendet werden, um eine Aktion nur dann zu vollziehen, wenn der DREF-Wert gültig definiert ist. Diese Fähigkeit erlaubt es, eine Steuerroutine zu schreiben, welche die dynamische Referenz verwenden kann, die jedoch nicht unterbrochen wird, wenn die dynamische Referenz während der Laufzeit ungültig ist. Selbstverständlich können andere Tests oder Befehle, welche das CONN-Attribut verwenden, verwendet werden, wie z. B. Verzweigungsbefehle, Halte- oder Abbruchbefehle etc. Ferner kann das CONN-Attribut jeden anderen gewünschten Wert (abgesehen von 0, -1 und größer als 0) annehmen, um den Erfolg oder das Versagen der Verbindung anzugeben.

- Das DRFV-Lese-/Schreibattribut (dynamischer Referenzfließkommawert) des dynamischen Referenzparameters wird verwendet, um das Lesen aus dem und Schreiben in das Feld, das durch das DREF-Attribut angegeben ist, als Fließkommawert oder ganzzahliger Wert zu ermöglichen. In einer Ausführungsform ist der Wert des DRFV-Attributs auf einen Maximalwert oder auf einen anderen spezifizierten Wert gesetzt, wenn das CONN-Attribut nicht 0 ist. Ferner verhindert in dieser Ausführungsform das DRFV-Attribut das Schreiben in das dynamisch referenzierte Feld, wenn das CONN-Attribut nicht 0 ist. Entsprechend wird das DRSV-Lese-/Schreibattribut (dynamischer Referenzstringwert) des dynamischen Referenzparameters verwendet, um das Lesen aus dem und Schreiben in das durch das DREF-Attribut als ein Stringwert spezifizierte Feld zu ermöglichen. In einer Ausführungsform ist das DRSV-Attribut so gesetzt, daß es ein leerer String ist, wenn das CONN-Attribut nicht 0 ist, und verhindert das Schreiben, wenn das CONN-Attribut nicht 0 ist. Dieses sind nützliche Attribute, da sie das Schreiben in das und Lesen aus dem dynamisch referenzierten Feld als entweder ein String- oder ein numerisches Feld oder als beides ermöglichen, wobei berücksichtigt wird, ob das dynamisch referenzierte Feld tatsächlich existiert oder gültig ist. Selbstverständlich könnten das DRFV- oder DRSV- oder andere spezifisch geschaffene Attribute verwendet werden, um Boolesche Werte oder Array-Werte (wie z. B. eine Gruppe von Werten, die in einem Array gespeichert ist) in dem durch das DREF-Attribut angegebenen Feld zu lesen und/oder zu schreiben.

- Das DRST-Attribut (dynamischer Referenzstatus) ermöglicht das Lesen des Statusattributs, das zu dem von dem DREF-Attribut spezifizierten Feld gehört. In bestimmten Steuereinrichtungs- oder Kommunikationsprotokollen, wie z. B. dem DeltaV- und dem Fieldbus-Protokoll, enthalten einige Parameter oder Felder einen Wert und einen Status, der den Status des Wertes angibt, z. B. ob der Wert gut, schlecht, unsicher, etc. ist. Das DRST-Attribut ermöglicht den Zugriff auf diesen Statuswert für einen dynamischen Referenzparameter. Das WRST-Attribut (Schreibstatus) liest den Schreibstatuswert des durch das DREF-Attribut bezeichneten Feldes. Dieser Status gibt den Erfolg der letzten Schreiboperation in das von dem DREF-Attribut bezeichnete Feld an und gibt Zugriff auf den Schreibstatus des dynamisch referenzierten Feldes.

- Selbstverständlich könnten nach Wunsch andere Attribute mit dem dynamischen Referenzparameter vorgesehen sein, um andere Operationen zu ermöglichen, wie z. B. das Lesen oder Schreiben in den Moduswert, Grenzwert oder in andere Statuswerte, die zu dem dynamisch referenzierten Feld gehören, oder die Durchführung jedes anderen Lese- oder Schreibvorganges in einem Attribut des dynamisch referenzierten Feldes. Entsprechend können die hierin identifizierten Attribute andere Namen oder Werte annehmen, um den Erfolg, das Versagen, etc. einer Verbindung oder eines Lese- oder Schreibvorganges anzugeben.

- Wenn ein dynamischer Referenzparameter im Kontext eines Einheitenmoduls verwendet wird, das heißt bei der Schaffung einer Einheitenphase, die eine dynamische Referenz enthält, werden Strings, die in das DREF-Attribut geschrieben werden, auf Aliasnamen untersucht, und eventuell vorhandene Aliasnamen werden auf der Basis der gegenwärtigen Alias-Auflösungstabelle für das Einheitenmodul ersetzt. Als Resultat können dynamische Referenzen geschaffen werden, um Felder zu spezifizieren, welche Aliasnamen verwenden, und diese Aliasnamen werden trotzdem noch aufgelöst, wenn die Einheitenphase aus der Phasenklasse geschaffen wird. Dies ermöglicht es, dynamische Referenzparameter in breiterem Umfang in Prozeßsteuerroutinen zu verwenden, auch wenn die dynamische Referenz nicht vor der Laufzeit aufgelöst werden kann oder während der Laufzeit auf der Basis eines Schreibvorganges in das DREF-Attribut des dynamischen Referenzparameters aufgelöst wird.

- Wenn SFC-Algorithmen verwendet werden, kann das Schreiben durch dynamische Referenzparameter auf mehrere Weisen erfolgen, in Abhängigkeit von der Gestaltung des SFC. Beispielsweise kann die Routine den gewünschten Wert als ein Statement in einem Schrittausdruck (unter der Annahme, daß die Bestätigung des Schreibvorganges, falls erforderlich, durch die Logik in späteren Teilen des SFC gehandhabt wird) zuordnen, die Routine kann eine Aktion des Impuls/Zuordnungstyps verwenden, mit der Bestätigung, einen Schreibvorgang auszuführen und anschließend zu pausieren, bis das WRST-Attribut einen anderen Wert annimmt als den Wert "in Betrieb" oder die Routine kann eine Aktion des

Nichtspeicherungs-/Zuordnungstyps verwenden, um den Wert wiederholt zu schreiben, bis der Übergangsausdruck erfaßt, daß der Wert erreicht wurde oder die Schrittzeit zu lang ist. Wenn somit SFC-Algorithmen verwendet werden, ist es bevorzugt, dynamische Referenzen durch die Verwendung von Aktionen des Impuls/Zuweisungstyps zu schaffen und zu verifizieren, mit einer Bestätigung, so daß der Aktionsausdruck den vollen dynamischen Referenzpfad auflöst und diesen dem DREF-Feld des entsprechenden dynamischen Referenzparameters zuweist (auf Modulebene oder Phasenebene), und so, daß der Bestätigungsausdruck prüft, ob der Wert des CONN-Attributs kleiner oder gleich 0 ist (verbunden oder Verbindung niemals möglich). Alternativ könnte bei dynamischen Referenzparametern, die gelesen werden sollen, der Wert des DRFV-Attributs gelesen und auf einen vernünftigen Wert (im Gegensatz zu dem MAX_Wert) geprüft werden und der Bestätigungszeitüberschreitungswert könnte auf eine kleine Zahl von Sekunden (beispielsweise fünf Sekunden) gesetzt werden.

Wenn ferner verschiedene dynamische Referenzparameter an demselben Punkt in einem Algorithmus geschaffen werden können, ist es bevorzugt, eine Aktion für jeden in demselben SFC-Schritt zu schaffen. Anschließend könnte ein einzelner Term in dem nachfolgenden Ausdruck des SFC-Übergangs, wie etwa: "RESOLVE_STEP/PENDING_CONFIRMS.CV" = 0" verhindern, daß der Algorithmus weiter abläuft, bis alle dynamischen Referenzparameter in ihrem Endzustand "stabilisiert" sind. Wenn der Algorithmus Verbindungen mit fehlendem Alias oder Verbindungen mit Alias IGNORIEREN handhaben muß, könnten nachfolgende Ausdrücke das CONN-Attribut des individuellen dynamischen Referenzparameters testen, um die Ausführung des Algorithmus zu leiten.

Sobald eine dynamische Referenz geschaffen wurde (das heißt das DREF-Attribut geschrieben wurde) und die dynamische Referenz verifiziert wurde (das CONN-Attribut ist 0), können die DRFV-, DRSV- und DRST-Felder geschrieben werden, um Werte in den referenzierten Parametern zu setzen. In kontinuierlichen Algorithmen (wie z. B. ein kontinuierlich ablaufender FAIL_MONITOR-Verbundblock) könnte die empfohlene Vorgehensweise für das Schreiben durch einen dynamischen Referenzparameter (der bereits geschaffen wurde) die Form annehmen:

```
IF("AUSLASS_POS.DRFV"!= <gewünschter Wert>) AND
("AUSLASS_POS.WRST"!= <in Ausführung Wert>) THEN
"AUSLASS_POS.DRFV" = <gewünschter Wert>;
```

wodurch kontinuierlich versucht wird, den referenzierten Parameter auf den gewünschten Wert zu steuern, bis dies in einer Weise erreicht wurde, die das Schreiben vermeidet, wenn der letzte Schreibversuch noch abläuft.

Es versteht sich, daß die Prozeßsteuerroutinen unter Verwendung von Aliasnamen, die vor der Laufzeit aufgelöst werden, und von indirekten Referenzen, wie z. B. dynamischen Referenzparametern, die während der Laufzeit aufgelöst werden, innerhalb jeder gewünschten Prozeßsteuerprogrammierung verwendet und implementiert werden können und in jedem Prozeßsteuersystem verwendet werden können, das jeden gewünschten Typ eines Prozeßsteuerkommunikationsprotokolls verwendet, und ferner verwendet werden können, um jede Art von Funktion bezüglich jeder Art von Einrichtung bzw. Einrichtungen oder Untereinheiten von Einrichtungen auszuführen. Prozeßsteuerroutinen, die indirekte Referenzierung verwenden, wie hierin beschrieben, werden vorzugsweise in Software implementiert, die beispielsweise in einer Steuereinrichtung oder einer anderen Prozeßsteuervorrichtung gespeichert ist. Diese Routinen können jedoch alternativ oder zusätzlich in Hardware, Firmware, etc. nach Wunsch implementiert werden. Wenn sie in Software implementiert sind, können die hierin erörterten Prozeßsteuerroutinen in jedem computerlesbaren Speicher gespeichert werden, wie z. B. auf einer Magnetplatte, einer Laserplatte oder einem anderen Speichermedium, in einem RAM oder ROM eines Computers oder einer Steuereinrichtung einer Anlageneinrichtung, etc. Entsprechend kann diese Software einem Benutzer oder einer Vorrichtung über jedes bekannte oder gewünschte Auslieferungsverfahren zugeführt werden, einschließlich beispielsweise über einen Kommunikationskanal, wie z. B. eine Telefonleitung, das Internet etc.

Patentansprüche

1. Prozeßsteuersystem zur Verwendung bei der Steuerung eines Prozesses, der mehrere Einheitenklassen hat, von welchen jede ein oder mehrere Einheitenmodule enthält mit:
einer Steuereinrichtung;
einem Speicher;
einer generischen Steuerroutine, die einen Aliasnamen verwendet; und
eine Alias-Auflösungstabelle für jedes der Einheitenmodule einer der Einheitenklassen, wobei jede Alias-Auflösungstabelle eine Aliasdefinition für den Aliasnamen hat;
wobei die generische Steuerroutine in dem Speicher (20, 22) gespeichert ist und dann, wenn die Steuerung eines bestimmten Einheitenmoduls (54, 64) erforderlich ist, die Steuereinrichtung (12) unter Verwendung der Alias-Auflösungstabelle (60) für das bestimmte Einheitenmodul eine diesbezüglich konkretisierte Version der generischen Steuerroutine für das bestimmte Einheitenmodul erstellt und das bestimmte Einheitenmodul steuert, indem die diesbezüglich konkretisierte Version der generischen Steuerroutine ausgeführt wird.
2. Prozeßsteuersystem nach Anspruch 1, dadurch gekennzeichnet, daß die generische Steuerroutine so ausgelegt ist, daß sie angewandt wird, um Einheitenmodule (54, 64) zu steuern, die zu zwei oder mehr ausgewählten Einheitenklassen gehören, und die Alias-Auflösungstabelle (60) für jedes der Einheitenmodule (54, 64) für jede der ausgewählten Einheitenklassen eine Aliasdefinition für den Aliasnamen hat.
3. Prozeßsteuersystem nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß es ferner eine Konfigurations-Workstation (14) enthält, die mit der Steuereinrichtung (12) in Kommunikationsverbindung steht, wobei die Workstation (14) einen Workstation-Speicher (20) enthält, der eine Konfigurationsroutine speichert, und einen Prozessor (24), der die Konfigurationsroutine ausführt, wobei die Konfigurationsroutine eine Prüfroutine enthält, die bestimmt, ob

die Alias-Auflösungstabelle (60) für jedes der Einheitenmodule (54, 64) für jede der ausgewählten Einheitenklassen eine gültige Aliasdefinition für den Aliasnamen enthält.

4. Prozeßsteuersystem nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß es ferner einen Indikator enthält, der zu der generischen Steueroutine gehört und die ausgewählten Einheitenklassen (54, 64) angibt, für welche die generische Steueroutine verwendet werden kann, um diesbezüglich konkretisierte Versionen der generischen Steueroutine zu schaffen.

5. Prozeßsteuersystem nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß es ferner einen Indikator für jedes der Einheitenmodule von mindestens einer Einheitenklasse enthält, der eine Einheitenmoduleeigenschaft angibt, die von Einheitenmodul zu Einheitenmodul variieren kann, wobei die generische Steueroutine mindestens zwei Steueralgorithmen enthält, die alternativ zu verwenden sind, um unterschiedliche Einheitenmodule basierend auf der Einheitenmoduleeigenschaft zu steuern, und wobei die Steuereinrichtung den Indikator für das bestimmte Einheitenmodul verwendet, um einen der mindestens zwei Steueralgorithmen auszuwählen, wenn die diesbezüglich konkretisierte Version der generischen Steueroutine für dieses bestimmte Einheitenmodul erstellt wird.

6. Prozeßsteuersystem nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, daß die Alias-Auflösungstabelle (60) für eines der Einheitenmodule (54, 64) eine Alias-Ignorieren-Definition für den Aliasnamen enthält, welche die Steuereinrichtung (12) veranlaßt, den Aliasnamen zu ignorieren, wenn die diesbezüglich konkretisierte Version der generischen Steueroutine für eines der Einheitenmodule ausgeführt wird.

7. Prozeßsteuersystem nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, daß die generische Steueroutine einen dynamischen Referenzparameter enthält, dessen Wert nach dem Erstellen der diesbezüglich konkretisierten Version der generischen Steueroutine für das bestimmte Einheitenmodul zugeordnet werden kann.

8. Prozeßsteuersystem nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, daß der dynamische Referenzparameter mehrere Attribute enthält, wobei eines der Attribute ein Referenzattribut ist, das einen Feldwert enthält, der ein Feld angibt, auf welches der dynamische Referenzparameter verweist.

9. Prozeßsteuersystem nach einem der Ansprüche 1 bis 8, dadurch gekennzeichnet, daß ein weiteres der Attribute ein Verbindungsattribut ist, das eine Angabe bietet, ob der Feldwert des Referenzattributes ein gültiges Feld ist.

10. Prozeßsteuersystem nach einem der Ansprüche 1 bis 9, dadurch gekennzeichnet, daß ein weiteres der Attribute ein Lese-/Schreibattribut ist, das in das von dem Referenzattribut angegebene Feld schreibt oder aus diesem liest.

11. Prozeßsteuersystem nach einem der Ansprüche 1 bis 10, dadurch gekennzeichnet, daß ein weiteres der Attribute ein Statusattribut ist, das einen Status liest, der zu dem durch das Referenzattribut angegebenen Feld gehört.

12. Prozeßsteuersystem nach einem der Ansprüche 1 bis 11, dadurch gekennzeichnet, daß das Statusattribut einen Schreibstatus liest, der einen Erfolg oder ein Versagen eines vorherigen Schreibvorganges in das von dem Referenzattribut angegebene Feld angibt.

13. Prozeßsteuersystem zur Verwendung bei der Steuerung eines Prozesses, der mehrere Einheitenklassen hat, von welchen jede eines oder mehrere Einheitenmodule enthält mit:

einer Steuereinrichtung;
einem Speicher;

einer generischen Steueroutine, die einen Aliasnamen verwendet und die so ausgelegt ist, daß sie zum Steuern von einem oder mehreren Einheitenmodulen angewandt werden kann, die zumindestens zwei ausgewählten Einheitenklassen gehören; und

eine Alias-Auflösungstabelle für jedes der Einheitenmodule jeder der ausgewählten Einheitenklassen, wobei jede Alias-Auflösungstabelle eine Aliasdefinition für den Aliasnamen hat;

wobei die generische Steueroutine in dem Speicher (20, 22) gespeichert ist und verwendet wird, um eine diesbezüglich konkretisierte Version der generischen Steueroutine für ein bestimmtes der Einheitenmodule (54, 64) unter Verwendung der Alias-Auflösungstabelle (60) für das bestimmte Einheitenmodul (54, 64) zu erstellen.

14. Prozeßsteuersystem nach Anspruch 13, dadurch gekennzeichnet, daß es ferner eine Konfigurations-Workstation (14) enthält, die mit der Steuereinrichtung (12) in Kommunikationsverbindung steht, wobei die Workstation (14) einen Workstation-Speicher (20) enthält, der eine Konfigurationsroutine speichert, und einen Prozessor (24), der die Konfigurationsroutine ausführt, wobei die Konfigurationsroutine eine Prüfroutine enthält, die bestimmt, ob die Alias-Auflösungstabelle (60) für jedes der Einheitenmodule (54, 64) für jede der ausgewählten Einheitenklassen eine gültige Aliasdefinition für den Aliasnamen enthält.

15. Prozeßsteuersystem nach Anspruch 13 oder 14, dadurch gekennzeichnet, daß es ferner einen Indikator enthält, der zu der generischen Steueroutine gehört und die ausgewählten Einheitenklassen (54, 64) angibt, für welche die generische Steueroutine verwendet werden kann, um diesbezüglich konkretisierte Versionen der generischen Steueroutine zu schaffen.

16. Prozeßsteuersystem zur Verwendung bei der Steuerung eines Prozesses, der mehrere Einheitenklassen hat, von welchen jede ein oder mehrere Einheitenmodule enthält mit:

einer Steuereinrichtung;
einem Speicher;

einem Indikator für jedes der Einheitenmodule mindestens einer der Einheitenklassen, der eine Einheitenmoduleeigenschaft angibt, die von Einheitenmodul zu Einheitenmodul variieren kann; und mit

einer generischen Steueroutine, die mindestens zwei Steueralgorithmen enthält, die alternativ zu verwenden sind, um unterschiedliche Einheitenmodule zu steuern, basierend auf der Einheitenmoduleeigenschaft der verschiedenen Einheitenmodule;

wobei die Steuereinrichtung den Indikator für ein bestimmtes Einheitenmodul verwendet, um einen der mindestens zwei Steueralgorithmen auszuwählen, um eine diesbezüglich konkretisierte Version der generischen Steueroutine für das bestimmte Einheitenmodul zu erstellen.

17. Prozeßsteuersystem zur Verwendung bei der Steuerung eines Prozesses, enthaltend:
eine Steuereinrichtung;

- einen Speicher; und
 einer Steuerroutine, die zur Steuerung mindestens eines Teils des Prozesses verwendet wird;
 dadurch gekennzeichnet, daß die Steuerroutine in dem Speicher (22) gespeichert wird, die Steuereinrichtung (12) eine ausführbare Version der Steuerroutine erstellt und den Teil des Prozesses durch Ausführen der ausführbaren Version der Steuerroutine steuert; und
 die Steuerroutine einen dynamischen Referenzparameter enthält, der mehrere Attribute einschließlich eines Referenzattributs hat, welches einen Feldwert enthält, der ein Feld angibt, auf welches der dynamische Referenzparameter verweist und das nach dem Erstellen der ausführbaren Version der Steuerroutine zugeordnet werden kann.
18. Prozeßsteuersystem nach Anspruch 17, dadurch gekennzeichnet, daß ein weiteres der Attribute ein Verbindungsattribut ist, das eine Angabe bietet, ob der Feldwert des Referenzattributs ein gültiges Feld ist.
19. Prozeßsteuersystem nach einem der Ansprüche 17 oder 18, dadurch gekennzeichnet, daß ein weiteres der Attribute ein Lese-/Schreibattribut ist, das in das von dem Referenzattribut angegebene Feld schreibt oder aus diesem liest.
20. Prozeßsteuersystem nach Anspruch 19, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein Stringwert lesbar oder schreibbar ist.
21. Prozeßsteuersystem nach Anspruch 19, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein numerischer Wert lesbar oder schreibbar ist.
22. Prozeßsteuersystem nach Anspruch 19, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein Boolescher Wert lesbar oder schreibbar ist.
23. Prozeßsteuersystem nach Anspruch 19, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein Arraywert lesbar oder schreibbar ist.
24. Prozeßsteuersystem nach einem der Ansprüche 17 bis 19, dadurch gekennzeichnet, daß ein weiteres der Attribute ein Statusattribut ist, das einen Status liest, der zu dem durch das Referenzattribut angegebenen Feld gehört.
25. Prozeßsteuersystem nach Anspruch 24, dadurch gekennzeichnet, daß das Statusattribut einen Schreibstatus liest, der einen Erfolg oder ein Versagen eines vorherigen Schreibvorganges in das von dem Referenzattribut angegebene Feld angibt.
26. Prozeßsteuersystem nach Anspruch 17, dadurch gekennzeichnet, daß ein zweites der mehreren Attribute ein Verbindungsattribut ist, das eine Angabe gibt, ob der Feldwert des Referenzattributs ein gültiges Feld ist und ein drittes der mehreren Attribute ein Lese-/Schreibattribut ist, das aus dem durch das Referenzattribut angegebenen Feld liest oder in dieses schreibt.
27. Prozeßsteuersystem nach Anspruch 26, dadurch gekennzeichnet, daß ein viertes der mehreren Attribute ein Statusattribut ist, das einen Status liest, der zu dem von dem Referenzattribut angegebenen Feld gehört.
28. Softwaresteuerkomponente zur Verwendung in einem Prozeßsteuersystem, das eine Steuereinrichtung hat, die einen Prozeß mit mehreren Einheitenklassen steuert, von welchen jede ein oder mehrere Einheitenmodule enthält, mit:
 einem computerlesbaren Speicher;
 einer generischen Steuerroutine, die in dem computerlesbaren Speicher gespeichert ist und einen Aliasnamen verwendet; und mit
 einer Alias-Auflösungstabelle für jedes der Einheitenmodule einer der Einheitenklassen, wobei jede Alias-Auflösungstabelle eine Aliasdefinition für den Aliasnamen hat;
 wobei die generische Steuerroutine so ausgelegt ist, daß sie von der Steuereinrichtung (12) in der Weise ausgeführt wird, daß dann, wenn die Steuerung eines bestimmten Einheitenmoduls (54, 64) erforderlich ist, die generische Steuerroutine von der Steuereinrichtung (12) verwendet wird, um eine diesbezüglich konkretisierte Version der generischen Steuerroutine für das bestimmte Einheitenmodul (54, 64) unter Verwendung der Alias-Auflösungstabelle (60) für das bestimmte Einheitenmodul zu schaffen.
29. Softwaresteuerkomponente nach Anspruch 28, dadurch gekennzeichnet, daß die generische Steuerroutine so ausgelegt ist, daß sie angewandt werden kann, um ein oder mehr Einheitenmodule (54, 64) zu steuern, die zu zwei oder mehr ausgewählten Einheitenklassen gehören, wobei die Alias-Auflösungstabelle (60) für jedes der Einheitenmodule für jede der ausgewählten Einheitenklassen eine Aliasdefinition für den Aliasnamen hat.
30. Softwaresteuerkomponente nach Anspruch 28 oder 29, dadurch gekennzeichnet, daß sie ferner einen Indikator enthält, der zu der generischen Steuerroutine gehört und die ausgewählten Einheitenklassen angibt, für welche die generische Steuerroutine verwendet werden kann, um diesbezüglich konkretisierte Versionen der generischen Steuerroutine zu schaffen.
31. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 30, dadurch gekennzeichnet, daß die Alias-Auflösungstabelle (60) für eines der Einheitenmodule (54, 64) eine Alias-Ignorieren-Definition für den Aliasnamen hat, welche die Steuereinrichtung (12) veranlaßt, den Aliasnamen zu ignorieren, wenn die diesbezüglich konkretisierte Version der generischen Steuerroutine für das eine der Einheitenmodule (54, 64) ausgeführt wird.
32. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 31, dadurch gekennzeichnet, daß sie ferner einen Indikator für jedes der Einheitenmodule mindestens einer Einheitenklasse enthält, der eine Einheitenmoduleeigenschaft angibt, die von Einheitenmodul zu Einheitenmodul variieren kann, wobei die generische Steuerroutine mindestens zwei Steueralgorithmen enthält, die alternativ so ausgelegt sind, daß sie zur Steuerung von verschiedenen Einheitenmodulen basierend auf den Einheitenmoduleigenschaften verwendet werden können, so daß die Steuereinrichtung den Indikator für das bestimmte Einheitenmodul verwenden kann, um einen der mindestens zwei Steueralgorithmen auszuwählen, wenn die diesbezüglich konkretisierte Version der generischen Steuerroutine für das bestimmte Einheitenmodul erstellt wird.
33. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 32, dadurch gekennzeichnet, daß die generische Steuerroutine einen dynamischen Referenzparameter enthält, dessen Wert nach dem Erstellen der instantiierten Version der generischen Steuerroutine für das bestimmten Einheitenmodul zugewiesen werden kann.

34. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 33, dadurch gekennzeichnet, daß der dynamische Referenzparameter mehrere Attribute enthält, wobei eines der Attribute ein Referenzattribut ist, das einen Feldwert enthält, der ein Feld angibt, auf welches der dynamische Referenzparameter verweist.

35. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 34, dadurch gekennzeichnet, daß ein weiteres der mehreren Attribute ein Verbindungsattribut ist, das eine Anzeige abgibt, ob der Feldwert des Referenzattributes ein gültiges Feld ist.

36. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 35, dadurch gekennzeichnet, daß ein weiteres der mehreren Attribute ein Lese-/Schreibattribut ist, das aus dem von dem Referenzattribut angegebenen Feld liest oder in dieses schreibt.

37. Softwaresteuerkomponente nach einem der Ansprüche 28 bis 36, dadurch gekennzeichnet, daß ein weiteres der mehreren Attribute ein Statusattribut ist, das einen Status liest, der zu dem durch das Referenzattribut angegebenen Feld gehört.

38. Softwaresteuerkomponente zur Verwendung in einem Prozeßsteuersystem, das eine Steuereinrichtung hat, die einen Prozeß mit mehreren Einheitenklassen steuert, von welchen jede ein oder mehrere Einheitenmodule enthält, mit:

einem computerlesbaren Speicher;

einer generischen Steueroutine, die in dem computerlesbaren Speicher gespeichert ist, welche einen Aliasnamen verwendet und die so ausgelegt ist, daß sie angewandt werden kann, um eines oder mehrere Einheitenmodule zu steuern, die zu mindestens zwei ausgewählten Einheitenklassen gehören; und mit

einer Alias-Auflösungstabelle für jedes der Einheitenmodule der ausgewählten Einheitenklassen, wobei jede Alias-Auflösungstabelle eine Aliasdefinition für den Aliasnamen hat;

wobei die generische Steueroutine so ausgelegt ist, daß sie von der Steuereinrichtung (12) verwendet wird, um eine diesbezüglich konkretisierte Version der generischen Steueroutine für ein bestimmtes Einheitenmodul (54, 64) unter Verwendung der Alias-Auflösungstabelle (60) für das bestimmte Einheitenmodul (54, 64) zu schaffen.

39. Softwaresteuerkomponente nach Anspruch 38, dadurch gekennzeichnet, daß sie ferner eine Konfigurationsroutine enthält, die eine Prüfroutine hat, welche bestimmt, ob die Alias-Auflösungstabelle (60) für jedes der Einheitenmodule (54, 64) für jede der ausgewählten Einheitenklassen eine gültige Alias-Definition für den Aliasnamen enthält.

40. Softwaresteuerkomponente nach Anspruch 38 oder 39, dadurch gekennzeichnet, daß sie ferner einen Indikator enthält, der zu der generischen Steueroutine gehört und die ausgewählten Einheitenklassen (54, 64) angibt, für welche die generische Steueroutine verwendet werden kann, um diesbezüglich konkretisierte Versionen der generischen Steueroutine zu schaffen.

41. Softwaresteuerkomponente zur Verwendung in einem Prozeßsteuersystem, das eine Steuereinrichtung hat, die einen Prozeß steuert, der mehrere Einheitenklassen hat, von welchen jede eines oder mehrere Einheitenmodule enthält, mit:

einem computerlesbaren Speicher;

einem Indikator für jedes der Einheitenmodule mindestens einer der Einheitenklassen, der eine Einheitenmoduleigenschaft angibt, die von Einheitenmodul zu Einheitenmodul variieren kann; und mit

einer generischen Steueroutine, die in dem computerlesbaren Speicher gespeichert ist und zwei Steueralgorithmen enthält, die alternativ zum Steuern von unterschiedlichen Einheitenmodulen auf der Basis der Einheitenmoduleigenschaft der verschiedenen Einheitenmodule verwendet wird;

wobei der Indikator so ausgelegt ist, daß er durch die Steuereinrichtung (12) für ein bestimmtes Einheitenmodul (54, 64) zu verwenden ist, um einen der beiden Steueralgorithmen auszuwählen, wenn die Steuereinrichtung (12) eine diesbezüglich konkretisierte Version der generischen Steueroutine für das bestimmte Einheitenmodul (54, 64) schafft.

42. Softwaresteuerkomponente zur Verwendung in einem Prozeßsteuersystem, das eine Steuereinrichtung hat, welche einen Prozeß steuert, mit:

einem computerlesbaren Speicher und

einer Steueroutine, die in dem computerlesbaren Speicher gespeichert ist und so angepaßt ist, daß sie von der Steuereinrichtung zur Steuerung mindestens eines Teils des Prozesses verwendet wird;

wobei die Steueroutine einen dynamischen Referenzparameter enthält, der mehrere Attribute hat, einschließlich eines Referenzattributs, das einen Feldwert enthält, der ein Feld angibt, auf welches der dynamische Referenzparameter verweist, und der nach dem Erstellen der ausführbaren Version der Steueroutine zugewiesen werden kann.

43. Softwaresteuerkomponente nach Anspruch 42, dadurch gekennzeichnet, daß ein weiteres der mehreren Attribute ein Verbindungsattribut ist, das eine Angabe gibt, ob der Feldwert des Referenzattributs ein gültiges Feld ist.

44. Softwaresteuerkomponente nach Anspruch 42 oder 43, dadurch gekennzeichnet, daß ein weiteres der mehreren Attribute ein Lese-/Schreibattribut ist, das aus dem von dem Referenzattribut angegebenen Feld liest oder in dieses schreibt.

45. Softwaresteuerkomponente nach Anspruch 44, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein Stringwert lesbar oder schreibbar ist.

46. Softwaresteuerkomponente nach Anspruch 44, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein numerischer Wert lesbar oder schreibbar ist.

47. Softwaresteuerkomponente nach Anspruch 44, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein Boolescher Wert lesbar oder schreibbar ist.

48. Softwaresteuerkomponente nach Anspruch 44, dadurch gekennzeichnet, daß das Lese-/Schreibattribut als ein Arraywert lesbar oder schreibbar ist.

49. Softwaresteuerkomponente nach einem der Ansprüche 42 bis 44, dadurch gekennzeichnet, daß ein weiteres der mehreren Attribute ein Statusattribut ist, das einen Status liest, der zu dem durch das Referenzattribut angegebenen

Feld gehört.

50. Softwaresteuerkomponente nach Anspruch 49, dadurch gekennzeichnet, daß das Statusattribut einen Schreibstatus liest, der den Erfolg oder ein Versagen eines vorgehenden Schreibvorganges in das von dem Referenzattribut angegebene Feld angibt.

51. Softwaresteuerkomponente nach Anspruch 42, dadurch gekennzeichnet, daß ein zweites der mehreren Attribute ein Verbindungsattribut ist, das eine Angabe gibt, ob der Feldwert des Referenzattributs ein gültiges Feld ist und ein drittes der mehreren Attribute ein Lese-/Schreibattribut ist, das aus dem durch das Referenzattribut angegebenen Feld liest oder in dieses schreibt.

52. Softwaresteuerkomponente nach Anspruch 51, dadurch gekennzeichnet, daß ein viertes der mehreren Attribute ein Statusattribut ist, das einen Status liest, der zu dem von dem Referenzattribut angegebenen Feld gehört.

Hierzu 3 Seite(n) Zeichnungen

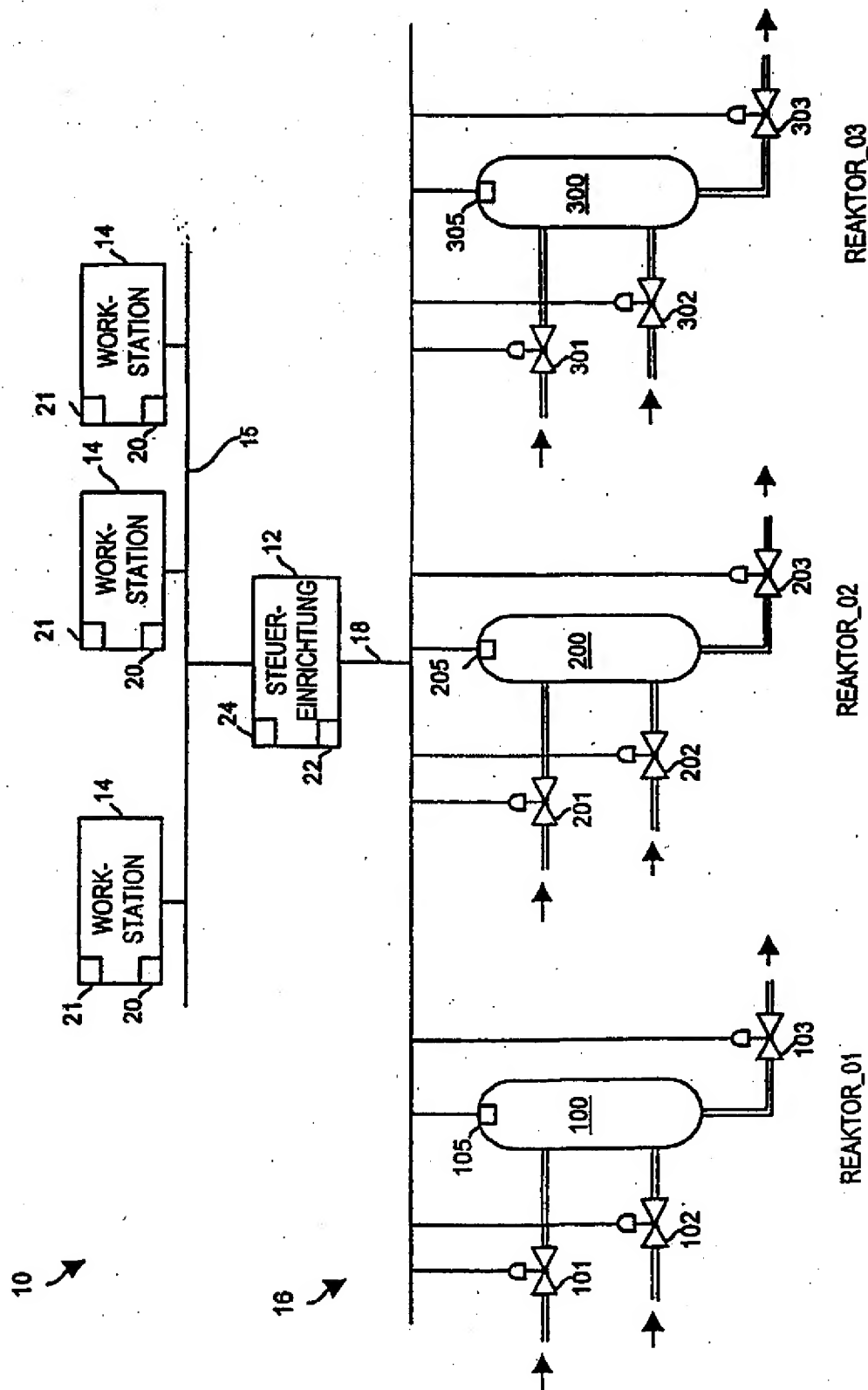


FIG. 1

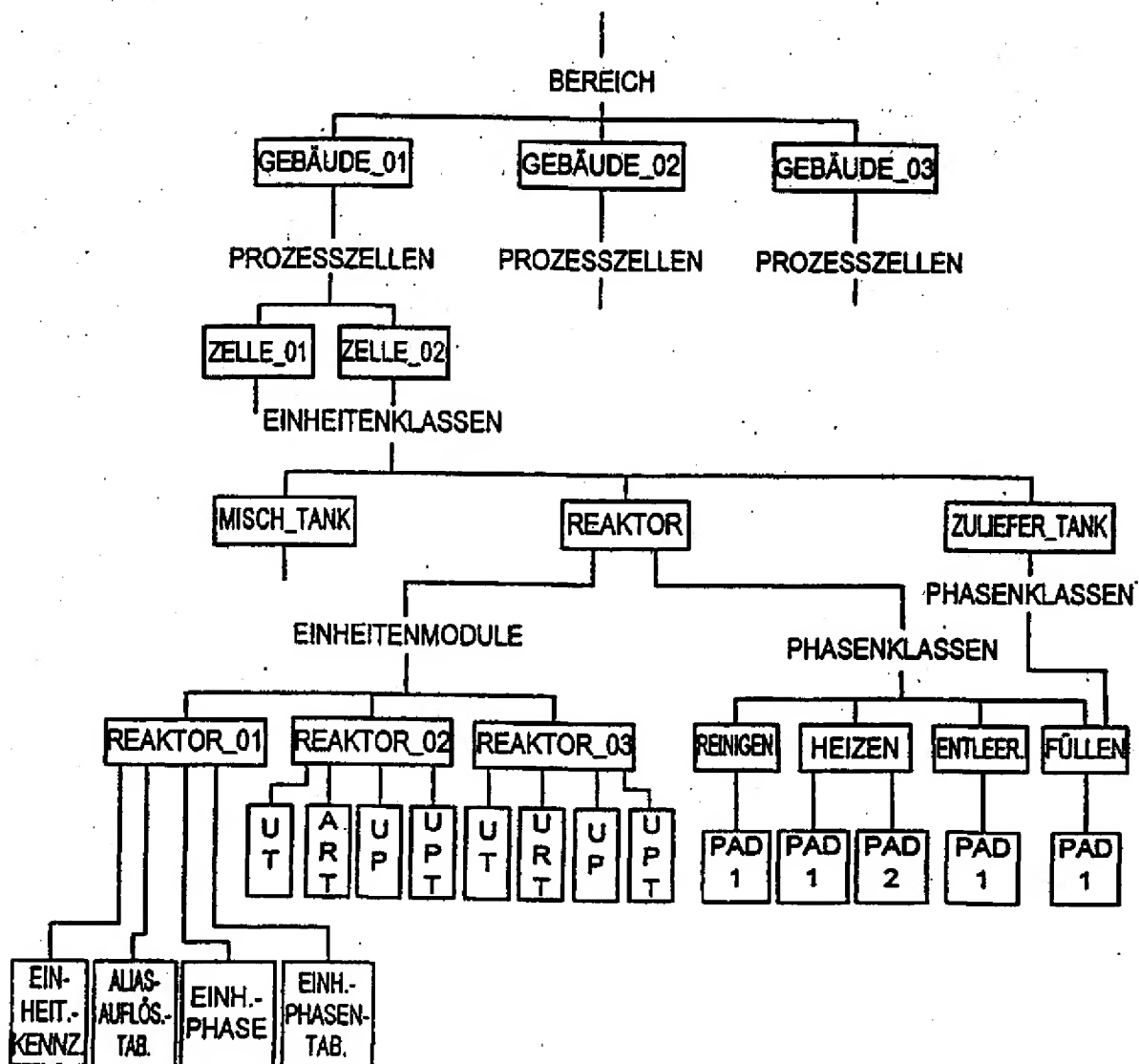


FIG. 2

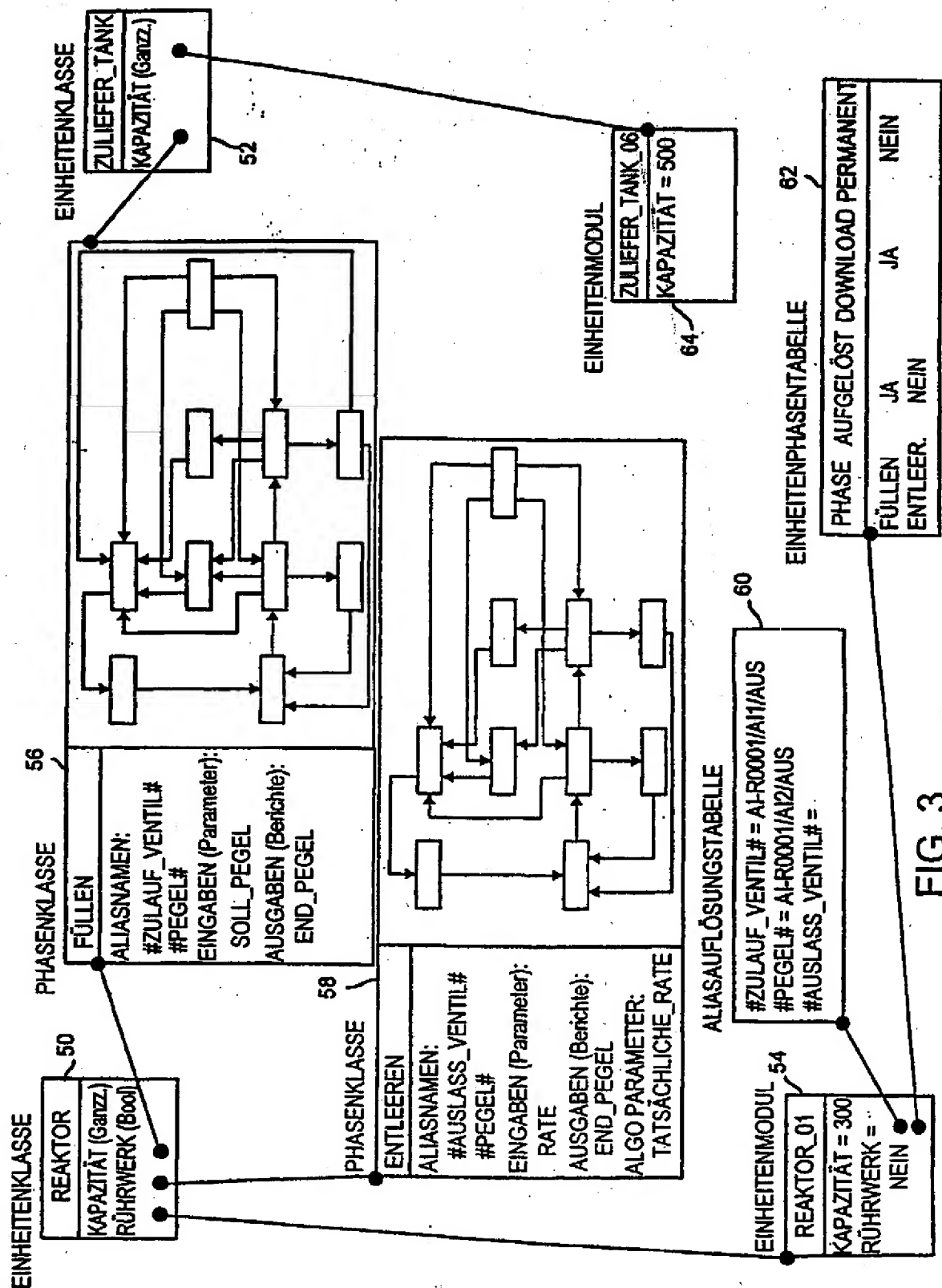


FIG. 3